TURING 图灵程序设计丛书

Peachpit Press



HTML5 and CSS3 Visual Quickstart Guide Seventh Edition

HTML5与CSS3基础教程

（第7版）

[美] Elizabeth Castro　Bruce Hyslop　著

望以文　译

- 风靡全球的畅销书最新版
- 讲述Web开发最新规范
- 零基础轻松掌握HTML5和CSS3



人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明



Elizabeth Castro

享誉世界的计算机畅销书作家，目前致力于帮助大家出版电子书、设计网页、运用CSS呈现网页和博客等。她擅长使用详实的步骤和精美的实例教大家快速实现具体的效果。有关她的更多信息，可访问其博客ElizabethCastro.com。



Bruce Hyslop

自1997年就开始从事网页开发，重点关注使用HTML、CSS和JavaScript进行网页开发和网站易用性的维护，并倡导最佳实践。Hyslop在加州大学洛杉矶分校进修部教授CSS课程，还著有*The HTML Pocket Guide* 。



望以文

毕业于中国人民大学信息资源管理学院，目前就职于百度。具有多年Web设计与开发经验，个人网站地址为http://weakow.com。



图灵程序设计丛书

HTML5 and CSS3
Visual Quickstart Guide, Seventh Edition

HTML5与CSS3基础教程
（第7版）

[美]　Elizabeth Castro　Bruce Hyslop　著

望以文　译

人民邮电出版社
北　京

图书在版编目（C I P）数据

HTML5与CSS3基础教程 : 第7版 / （美）卡斯特罗
(Castro,E.)，（美）希斯洛普（Hyslop,B.）著 ; 望以
文译. -- 北京 : 人民邮电出版社, 2013.1
（图灵程序设计丛书）
书名原文: HTML5 and CSS3:Visual Quickstart
Guide,Seventh Edition
ISBN 978-7-115-30027-0

Ⅰ. ①H… Ⅱ. ①卡… ②希… ③望… Ⅲ. ①超文本
标记语言—程序设计②网页制作工具 Ⅳ. ①
TP312②TP393.092

中国版本图书馆CIP数据核字(2012)第281881号

内 容 提 要

本书是讲解HTML和CSS入门知识的经典畅销书，如今已经更新到第7版，介绍时下相当热门的HTML5和CSS3。虽然HTML5和CSS3还没有成为W3C的推荐标准，但主流浏览器已经能够很好地支持它们的大部分特性。

本书不仅介绍了文本、图像、链接、列表、表格、表单、多媒体等网页元素，也介绍了如何为网页设计结构、布局，添加动态效果、格式化等形式，此外还涉及调试和发布、聚合和吸引访问等。书中详细讲解了视频、音频及其他新增特性，从零开始教会读者创建渐进增强的普适性网站。书中提供了大量代码示例并附上代码实现的屏幕截图，配套网站上列出了完整的示例代码及更多实例。

本书适合网站设计新手和专业开发人员学习参考。

图灵程序设计丛书
HTML5与CSS3基础教程（第7版）

◆ 著　　　[美] Elizabeth Castro　Bruce Hyslop
　译　　　望以文
　责任编辑　朱　巍
　执行编辑　刘美英
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
　邮编　100061　　电子邮件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京　　　　　印刷
◆ 开本：800×1000　1/16
　印张：24.5　　彩插 8
　字数：579千字　　　　2013年1月第1版
　印数：1－4 000册　　　2013年1月北京第1次印刷
著作权合同登记号　图字：01-2012-1660号
ISBN 978-7-115-30027-0

定价：59.00元
读者服务热线：(010)51095186转604　印装质量热线：(010)67129223
反盗版热线：(010)67171154

版权声明

译 者 序

2007年8月，图灵公司出版《HTML XHTML CSS基础教程（第6版）》。那本书正是当年我学习Web设计与开发的入门书。不过，当时的我完全不曾想到自己会成为该书下一版的译者。五年过去了，今非昔比，互联网的发展日新月异，而作为网页基础的HTML与CSS技术也在不断“变革”。HTML已更新至HTML5，CSS也改进为CSS3，虽然两者还未成为网页构建新标准，但这只是时间的问题，况且主流浏览器对绝大多数新特性已经支持良好。如今，这套经典教程也更名为《HTML5与CSS3基础教程（第7版）》，重出江湖，全面讲解时下炙手可热的HTML5与CSS3技术。

HTML（超文本标记语言）是少有的在名称中包含“语言”一词的常见语言，然而讽刺的是，很多人都认为HTML算不上是一门计算机语言（其中隐含的意思则是HTML不需要花费工夫学习）。CSS（层叠样式表）则是另一种与网页有关的语言，它同HTML一样简单易学，在很多人眼里也是不值一学的技术。尽管HTML和CSS学起来都不难，但它们毕竟是万维网的基础，是每一名Web设计和开发人员都要掌握的技术。实际上，HTML和CSS的简单主要体现在模式单纯，入门容易。很多人深知这一特点，却忘了它们都是知识点繁杂，需要极其注重细节的技术。最典型的例子莫过于对浏览器兼容性的追求。万维网的普适特性使得用户的浏览环境注定不会单一，近年来移动互联网的迅猛发展又加剧了这种多样性。有经验的Web设计和开发人员都知道，要确保网页对不同浏览器的兼容，着力提升网站的可用性，可不是一件容易的事。

这本书是广受欢迎的HTML与CSS经典入门图书，迄今更新至第7版，第1版可追溯到万维网兴起之初。这本书从最基本的HTML标记和CSS属性讲起，全面、透彻地介绍了HTML与CSS的方方面面。值得一提的是，本书独特的双栏版式恰当地将代码输入步骤、示例代码块和效果图示组织在一起，很适合HTML和CSS这种主题繁多但形式统一的教学。此外，在本书历版中作为附录的HTML标记和CSS属性完整列表向来都是初学者的重要参考资料，它们在这一版中又作了更新，相当有价值。

与市面上大多数介绍HTML5和CSS3的书不同，本书并非只讲解新技术，而是紧密结合HTML5和CSS3的最新规范和最佳实践，从基础讲起，构建了一套全面的知识体系。HTML5和CSS3的出现为Web开发引入了一些新的实践方法，也让一些做法变得过时。对于初学者来说，这本书可以让他们从一开始就学到“正确的”做法，而不必先学到陈旧的知识，日后再作修正。在市面上同类优质图书相对匮乏的情况下，这本书的出现也犹如雨露甘霖，定能为Web设计与开发初学者带好启程之路。

最后，借此机会，感谢图灵公司对我的信任，感谢傅志红、楼伟珊、刘美英等编辑的细心工作，他们的耐心指导和建议给了我很大帮助。感谢我的父母，他们的支持与宽容总是我开展任何艰苦工作的情感依托，而他们的健康与幸福则是我最大的冀望。

前　言

无论你是刚开始涉足网站制作，还是曾经做过网站、而今又想让自己的技能与时俱进，都会对这个领域振奋人心的进展感同身受。

在过去的几年里，为网页编写代码和添加样式的方式、浏览网页所用的浏览器以及使用浏览器的设备都发生了明显的变化。曾经，我们只能通过台式机或笔记本浏览万维网，而如今我们可以通过很多设备访问万维网：手机、平板电脑，当然也包括笔记本、台式机，以及更多别的设备。

这是意料之中的事，因为万维网始终秉持消除边界的宗旨，无论在城市还是在乡村，通过任何能访问万维网的设备，任何人都可以自由地分享和获取信息。总之，万维网的宗旨是普适性。万维网的覆盖范围不断扩大，过去，通信技术还未曾普及到农村，而现在却已经实现了。

万维网的伟大之处还在于每个人都可以自由地创建和发布网站。本书正是没有任何HTML和CSS知识的建站初学者的理想教程，将指导读者创建和发布网站。书中内容结构清晰，浅显易懂，将一步一步地教会你如何创建网页。总之，本书是开发人员手头案边的必备指南，若想了解某个主题更多信息，通过查找目录可以直接跳到相关页面。

HTML 和 CSS 简介

万维网成功的根基，是一种基于文本的标记语言——HTML，它简单易学，并且能被任何带有基本Web浏览器的设备识读。每个网页都至少用一点儿HTML，否则也就不能称为网页了。

随着学习的深入细致，你会了解到，HTML用于定义内容的含义，而CSS用于定义内容和网页如何显示。HTML页面和CSS文件（样式表，style sheet）都是文本文件，因此很容易编辑。在下面的“如何使用本书”一节，你将看到一些HTML和CSS的代码片段。

从第1章起，我们开始学习基本的HTML页面；从第7章起学习用CSS定义页面样式。关于本书各章内容概述及主题汇总，请参见“本书涉及内容”。

1. 什么是HTML5

了解一些有关HTML起源的基础知识对于理解HTML5是有帮助的。HTML诞生于20世纪90年代初，用于详细规定少量构建网页的元素。这些元素中的大多数都用于描述网页内容，如标题、段落、列表等。随着更多元素的引入以及对语法规则本身的调整，HTML这门语言的版本号也在更新。当前最新的版本便是HTML5。

HTML5是HTML早期版本的自然延续，它尽可能地满足当前网站和未来网站的需求。它从以前的版本中继承了大部分特性，这意味着，如果你在HTML5出现之前写过HTML，那么你已经知道很多关于HTML5的知识了。这也意味着，HTML5的大部分内容都可以兼容新旧浏览器，向后兼容是HTML5的一项重要设计原则（参见www.w3.org/TR/html-design-principles/）。

HTML5还增加了不少新功能。很多新功能都很简单，比如用于描述内容的辅助元素（如article、section、figure等）。还有一些用于帮助创建强大的Web应用程序的新功能则非常复杂。只有牢牢掌握了创建网页的技能，才能去学习HTML5更复杂的功能。HTML5还引入了原生的音频和视频播放功能，这在本书中也会讲到。

2. 什么是CSS3

HTML诞生几年后才出现CSS的第一个版本。CSS于1996年正式推出。同HTML5与其早期版本的关系一样，CSS3也是CSS早期版本的自然延续。

CSS3比CSS早期版本更为强大，它引入了大量的视觉效果，如外阴影、文字阴影、圆角、渐变等。（关于CSS3涵盖的详细信息，请参见“本书涉及内容”。）

3. Web标准与规范

你可能在想，是谁第一个创造了HTML和CSS，又是谁在持续地发展它们。由万维网和HTML的发明者Tim Berners-Lee主持的万维网联盟（W3C）是负责带领Web标准发展的组织。规范（specification，缩写为spec）是定义HTML、CSS等语言的参数的文档。也就是说，规范对规则进行了标准化。要了解W3C的活动，请登录www.w3.org（见图1）。

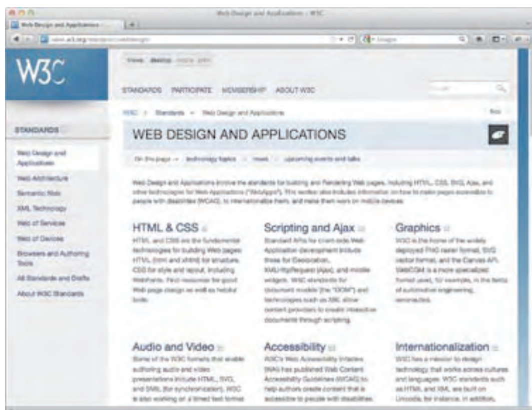


图1 W3C的网站是业内Web标准规范的主要信息来源

由于各种原因，另一个组织，Web超文本应用技术工作组（WHATWG，www.whatwg.org），也在开发HTML5规范。W3C将WHATWG的工作纳入了其正在开发的规范的正式版本之中。

利用已经完成的标准，我们可以根据一套成形规则创建网页，而Chrome、Firefox、Internet Explorer（IE）、Opera和Safari等浏览器也能根据这套规则显示这些页面。（整体上，浏览器对标准的支持是良好的。IE的早期版本，尤其是IE6，则存在一些问题。）

经过几个阶段的发展，规范最终会被确定下来，成为推荐标准（Recommendation，www.w3.org/2005/10/Process-20051014/tr）。

部分HTML5和CSS3规范仍未最终敲定，但这并不意味着你不能使用这些规范。标准化的过程需要经历一些时间（准确地说需要数年）。浏览器会在规范成为推荐标准之前很久便实现其功能，这也是规范发展过程的反映。因此，浏览器已经包括了大量HTML5和CSS3的功能，尽管它们还没有成为推荐标准。

总体上，本书涉及的功能都是规范中较为明确规定的，因此它们在推荐标准推出之前发生改变的可能性极小。很多HTML5和CSS3功能已经实际使用一段时间了，你完全可以放心使用。

渐进增强：一种最佳实践

我在前言的开头处就讲到了万维网的普适性——万维网上的信息应该能被所有人访问。渐进增强（progressive enhancement）能帮助你构建具有普适性的网站。这不是一门语言，而是一种建站方法，它由Steve Champeon于2003年提出（http://en.wikipedia.org/wiki/Progressive_enhancement）。

这个想法很简单，但也很强大：用所有人都能访问的HTML内容和行为开始构建网站（参见图2）；对同一个页面，用CSS加入你的设计（参

见图3），用JavaScript添加额外的行为。这些CSS和JavaScript通常是从外部文件加载进来的（后文会讲到如何做到这些）。



图2　一个基本的HTML页面，未应用任何自定义的CSS。这个页面可能并不是很好看，但信息都是可访问的，这一点非常重要。即使是20多年前万维网刚产生时的浏览器，都能显示这个页面；最早配有Web浏览器的手机也能显示。此外，屏幕阅读器（可以为视障访问者读出网页的软件）也能轻松地处理这个页面



图3　同一个页面在支持CSS的浏览器中的显示效果。信息是相同的，只是呈现的方式不同。使用功能更强的设备和浏览器访问这个页面的用户得到了增强的体验

这样做的结果就是，那些只能访问基本页面的设备和浏览器会得到简化的、默认的体验，而那些能够浏览更健壮的网站的设备和浏览器将看到增强的版本。不必要求网站对所有人来说体验都是一样的，关键是网站的内容是可访问的。本质上，渐进增强背后的涵义是人人共赢。

这本书将会讲解如何建立渐进增强的网站，但在讲述的过程中不会总是把这个词明确地说出来。遵循贯穿全书的最佳实践，自然就能做到渐进增强。

不过，第12章和第14章明确强调了渐进增强。如果你想了解以下内容，可以先阅读这两章：如何用渐进增强的原则构建其布局能适应设备屏幕大小和浏览器功能的网站，如何让旧的浏览器显示简化的设计而现代浏览器显示包含CSS3效果的增强版本。

渐进增强是构建人人都能访问的网站的关键。

目标读者

本书假定读者没有任何做网站的知识。因此，从这个意义上说，这本书适合零起点初学者。你将从最基础的HTML和CSS知识学起。在这个过程中，你也会学到HTML5和CSS3的新特性，尤其是那些设计师和开发者在日常工作中经常用到的特性。

不过，即使你对HTML和CSS很熟悉，依然可以阅读本书，特别是如果你想快速了解HTML5、CSS3的大量最新特性以及最佳实践的话。

1. 本书涉及内容

为了给读者带来尽可能多的材料，本书英文原版比上一版多约125页的内容。（第1版出版于1996年，只有176页。）此外，我们对上一版几乎每一页都进行了大量更新（甚至完全重写）。总之，第7版是一次重大的修订。

全书各章是按照如下方式进行组织的。

- 第1章至第6章及第15章至第18章讲解创建HTML页面的原则和你用得上的HTML元

素，并清楚地说明什么时候该使用，以及如何使用这些元素。

- 第7章至第14章讲解CSS，从创建第一条样式规则开始，一直到运用CSS3增强视觉效果。
- 第19章演示如何向页面添加预先写好的JavaScript。
- 第20章讲解在把页面放到万维网上之前如何对其进行测试和调试。
- 第21章讲解如何保护自己的域名，以及如何把网站放到万维网上让其他所有人都能看到。
- 附录A和附录B，其中附录A列出了HTML全局属性、HTML元素及其属性；附录B列出了CSS属性及其值。这些内容可供读者快速查阅。

部分主题内容如下。

- 创建、保存、编辑HTML和CSS文件。
- 编写语义化HTML的含义及其重要性。
- 如何把页面内容（即HTML）和呈现（即CSS）分开（这是渐进增强的关键要素）。
- 通过一种有含义的方式，使用那些存在多年的HTML5和新增的HTML元素对内容进行结构化。
- 使用ARIA地标角色和其他好的编码实践提升网站的可访问性。
- 在页面中添加图片，并针对万维网对其进行优化。
- 从一个网页链接到另一个网页，或从页面的一部分链接到另一部分。
- 为文字添加样式（大小、颜色、粗体、斜体等）；添加背景颜色和背景图片；实现浮动的多栏布局，这种布局可根据不同的屏幕尺寸缩小或放大。
- 充分利用CSS3中新的选择器。这些新的选择器提供了更多定位样式的方法。
- 学习为移动设备访问者准备网页。
- 基于许多响应式Web设计原则（其中的一些影响了CSS3媒体查询），为所有用户创建单一网站，不管他们使用的是手机、平板电脑、笔记本、台式机，还是其他能访问万维网的设备。
- 使用`@font-face`为页面添加自定义的Web字体。
- 应用CSS3效果，如透明度、背景alpha透明、渐变、圆角、外阴影、内阴影、文字阴影以及多背景图片。
- 创建用于获取访问者输入内容的表单，包括使用一些HTML5中新增的输入类型。
- 使用HTML5的`audio`和`video`元素向页面插入媒体。

当然，还有很多。

这些主题都配有众多代码示例，说明如何基于业内最佳实践来实现HTML5和CSS3的各个特性。

2. 本书未涉及内容

唉，尽管比起上一版，我们新增了许多内容，但还是不得不舍弃很多其他的HTML和CSS主题。

除了几处例外，我们坚持把那些极少用到、仍可能变化、缺乏广泛浏览器支持或者需要JavaScript知识的主题，以及一些高级主题排除在本书范围以外。

本书没有涉及下列主题。

- HTML5的`details`、`summary`、`menu`、`command`和`keygen`元素。
- HTML5的`canvas`元素，它让你可以使用JavaScript绘制图像（甚至创建游戏）。
- 需要JavaScript知识或者与HTML5新增的语义化元素没有直接关联的HTML5 API和其他高级特性。
- CSS精灵。这项技术可以将多张图片拼合成一张图片，这有助于减少页面对需要加载的资源的请求数。更多信息请参见www.bruceontheloose.com/sprites/。

- CSS图片替换。这项技术通常与CSS精灵一起使用。更多信息请参见www.bruceon-theloose.com/ir/。
- CSS3变换、动画和过渡效果。
- CSS3中新的布局模型。

如何使用本书

本书几乎所有章节都有演示真实用法的代码示例（参见图4和图5）。通常情况下，随后还会展示这些代码对应的页面在浏览器中显示的屏幕截图，参见图6。

```
...
<body>
<header role="banner">
    ...
       <nav role="navigation">
          <ul class="nav">
             <li><a href="/" class="current">home</a></li>
             <li><a href="/about/">about</a></li>
             <li><a href="/resources/">resources</a></li>
             <li><a href="/archives/">archives</a></li>
          </ul>
       </nav>
    ...
</header>

...

</body>
</html>
```

图4　你将在本书很多地方看到HTML代码片段，对应的部分会被突出显示。省略号（…）表示出于简洁目的而省略的额外代码或内容。通常，省略的部分可在其他代码示例中看到

```
/* 站点导航 */
.nav li {
    float: left;
    font-size: .75em; /* makes the
    → bullets smaller */
}

.nav li a {
    font-size: 1.5em;
}

.nav li:first-child {
    list-style: none;
    padding-left: 0;
}
```

图5　如果示例有CSS代码，它们会单独显示，对应的部分也会突出显示

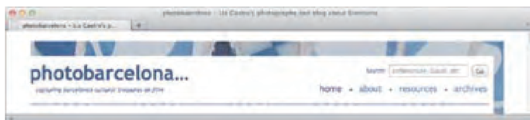


图6　用于演示代码如何影响页面的一种或多种浏览器的屏幕截图

大多数屏幕截图用的都是当下最新版本的Firefox。不过，这并不意味着，与其他浏览器相比，我们更推荐Firefox。本书的代码示例在最新版本的Chrome、Internet Explorer、Opera和Safari中看起来都是相似的。你将在第20章了解到，应该在不同的浏览器中测试页面，因为你无从知道访问者使用的是什么浏览器。

代码和屏幕截图中都有对相应HTML元素和CSS属性的描述，以便于对示例的背景进行说明，并加强你对它们的理解。

在很多情况下，你会发现只要理解说明和代码示例就可以使用HTML和CSS特性了。但如果你需要这些特性使用方法的明确指导，也可以阅读书中列出的详细步骤讲解。

最后，大部分章节都有一些提示，用于提供额外的用法信息、最佳实践、对本书相关部分的引用、对其他资源的链接等。

本书约定

本书遵循以下约定。

- HTML一词通常泛指这门语言。HTML5则用于指代这一特定版本的HTML，例如在讨论HTML早期版本所没有的HTML5新特性的时候。CSS（泛指）和CSS3（特指）的用法与之类似。
- 需要你自己输入值的占位符的文字或代码以斜体显示。大多数占位符出现在步骤讲解中。例如，“输入#*rrggbb*，其中*rrggbb*是颜色的十六进制代码。”
- 你需要准确输入的代码（即HTML和CSS代码）使用等宽字体（如`this font`）。
- 代码图中的箭头（→）代表延续上一行的代码（这一行代码由于篇幅原因而被折断了），如图5所示。箭头并非代码的一部分，不需要输入。你应该连续地输入这一行代码，就好像它没有被折断。
- 对于需要定义的词，第一次出现时会用楷体显示。
- IE是对Internet Explorer的简称。例如，IE9与Internet Explorer 9同义。
- 在浏览器的版本号后面出现的加号（+）指的是该版本之后的各个版本。例如，Firefox 8+指的是Firefox 8.0及其后所有的版本。

本书配套网站

本书网站（www.bruceontheloose.com/htmlcss/）上有本书的目录、本书涉及的（以及一些额外的不适合放入书中的）完整代码示例、引用资源的链接（以及一些额外资源的链接）、写作时用做参考的信息、勘误表，附录A和附录B，等等。

www.bruceontheloose.com/htmlcss/examples/列出了所有代码示例。你可以在网页上查看这些代码，也可以把它们下载到你的电脑里——所有HTML和CSS文件你都可以拿到。

有时，我会在代码里插入一些额外的注释，用以解释对应代码的含义和用法。限于篇幅，本书大量的代码示例都有所删减，但配套网站列出了完整的代码。读者可随意使用这些代码，并对它们进行修改，使之满足自己项目的需要。

下面是本书网站一些重要页面的URL。

- 主页：www.bruceontheloose.com/htmlcss/
- 代码示例：www.bruceontheloose.com/htmlcss/examples/
- 附录A：www.bruceontheloose.com/ref/html/
- 附录B：www.bruceontheloose.com/ref/css/

希望该网站对你有用。

致　谢

撰写致谢可谓是写书过程中最严峻的挑战之一，因为你需要确保将对参与本书出版的所有人的感激之情合理地表达出来。本书的出版得到了许多人的大力支持，是他们不懈努力的成果，凝聚了他们的美好愿望。我希望能公平地对待他们，请允许我在此表达对他们深深的谢意。

将最诚挚的感谢献给以下各位。

Nancy Aldrich-Ruenzel和Nancy Davis，感谢他们对我的信任，本书是Peachpit的重点图书，能够撰写本书，我荣幸之至。

Cliff Colby，感谢他的推荐以及对我的信任。Cliff耐心、灵活、幽默、富有主见，与他的多次交流让我受益匪浅。

Robyn Thomas，感谢他付出艰辛的努力确保了本书按时出版，Robyn就本书内容与我们进行了多次探讨，提出了大量建议，并进行了周密的编辑，这对我们来说是莫大的帮助。

Michael Bester，感谢他提供准确的反馈和建议，指出技术错误和疏漏，帮助我们将正确的内容呈现给读者。此外，与他在另一本书上的合作也很愉快。

Chris Casciano，同样感谢他的技术意见、建议和重要反馈。非常感激他在最后几周加入我们，这是我们的荣幸。

Cory Borman，感谢他对本书出版的专业监督，感谢他为本书创建图表，以及他带给我们的幽默。

Scout Festa，感谢他仔细地校对语法、标点，打磨语言，确保图和章节引用的准确性，总的来说就是对本书内容进行全面修饰和润色。

David Van Ness，感谢他细致的页面排版工作，感谢他对细节的熟稔和关注。

Nolan Hester，感谢他对排版后书页的专业校对。

Valerie Haynes Perry，感谢他创建了一个高效的索引，可供读者反复查阅。

Peachpit的市场、销售等工作人员，感谢他们的幕后工作让本书获得成功。

我们家人和朋友，感谢他们关注我的进展，并不时地让我从写作中解放出来休息一下。我还要特别感谢那些恐怕已厌倦我常说无法参加聚会，却仍不停地邀请我的朋友。

Robert Reinhardt，一如既往，感谢他让我接触写书，感谢他对我着手写作此书的指导。

Web社区，感谢你们的创新和对知识的分享，从而让他人获益（我在全书多处引用了你们的内容）。

读者朋友，感谢你们对学习HTML和CSS的兴趣，感谢你们选择本书（我知道还有很多书可供选择），我希望本书能够对你们有所帮助。

非常感谢以下特约撰稿人。感谢你们的工作，正是因为你们的努力，读者才拥有一本更有价值的书。我还要向Erik Vorhes表示歉意，因为我们无法将附录A和附录B放到书里[①]。从本书网站上

① 中文版加入了附录A和附录B。——编者注

看到它们的读者一定会感谢你的工作。

本书的特约撰稿人包括以下几位（按姓氏首字母顺序排列）。

Scott Boms（第14章）

Scott是一位屡获殊荣的设计师、作家和演讲家，在他超过15年的Web生涯中，曾与PayPal、汇丰银行、现代、DHL、XM Radio、*Toronto Life*杂志及Masterfile合作过。当他远离计算机时，你可能会发现他在用宝丽来摄影，在他的乐队George里打鼓，或是与他出色的妻子和两个孩子共享美好时光。他的Twitter账户是@scottboms。

Ian Devlin（第17章）

Ian Devlin是一位爱尔兰Web开发者、博主和作家，他喜欢编写代码以及撰写关于HTML5、CSS3等新生Web技术的图书。除了前端开发，Ian还会构建基于.NET、PHP等后端技术的解决方案。他最近撰写的一本书为*HTML5 Multimedia: Develop and Design*（Peachpit，2011）。

Seth Lemoine（第5章和第16章）

Seth Lemoine是亚特兰大软件开发者和教师。十几年来，他参与了一些富有挑战性的项目，用到了从HTML、JavaScript、CSS到Objective-C、Ruby等各种技术。无论是寻找向学生教授HTML5和CSS的创新性方法，还是用他的炒锅探索川菜食谱，他都展现了很高的创造力。

Erik Vorhes（附录A和附录B）

Erik Vorhes在VSA Partners公司从事Web开发，同时也是Typedia（http://typedia.com/）的总编。他在芝加哥生活和工作。

Brian Warren（第13章）

Brian Warren是费城Happy Cog的高级设计师。写作和设计工作之余，他会花时间陪家人玩耍，听音乐，酿啤酒。他的博客（断断续续地写了一些）：http://begoodnotbad.com。

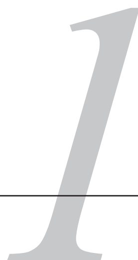
最后，特别感谢Elizabeth Castro。她在15年前完成了第1版，并通过其后的版本让众多读者受益。多年以来，她的教学风格获得了成千上万读者的共鸣。能够参与本书写作，我备感荣幸。不管是对于书中的内容还是对于读者，写作过程中我都做到了精雕细琢，希望读者批评指正。

Bruce

目　　录

第 1 章

网页的构造块



本章内容

- 基本 HTML 页面
- 语义化 HTML：有含义的标记
- 标记：元素、属性和值
- 网页的文本内容
- 链接、图像和其他非文本内容
- 文件名
- URL
- 要点回顾

尽管网页变得越来越复杂，但是其底层结构依然相当简单。你应当知道的第一件事就是，创建网页都离不开 HTML。你即将了解到，HTML 会包围内容并说明这些内容表示什么，Web 浏览器则会将 HTML 包着的内容呈现给用户。

一个网页主要包括以下三种成分。

- 文本内容（text content）：在页面上让访问者了解页面内容的纯文字，比如关于你的业务、产品、家庭度假等内容。
- 对其他文件的引用（references to other files）：这些文件加载图像、音频、视频、SVG[①]文件等，指向其他 HTML 页面和资源，以及样式表（用于控制页面的布局）和 JavaScript 文件（用于为页面增加行为）。
- 标记（markup）：对文本内容进行描述并使引用正确地工作。（HTML 一词中的字母 M 就代表标记。）

需要注意的是，网页的这些成分都仅由文本构成。这意味着网页可以保存为纯文本格式，可以在任何平台（无论是台式机、手机、平板电脑还是其他平台）上用任何浏览器查看，从而保证万维网的普适性。同一页面在不同的设备上看起来可能并不一样，不过这没关系。重要的是，第一步要让内容对所有用户都是可访问的，而 HTML 可以做到这一点。

除了上述三种主要成分，网页还包括一些其他 HTML，它们提供关于页面本身的信息，比如网页内容的主要语言（英语、法语等）、字符编码（通常为 UTF-8）等。这些 HTML 主要是为浏览器和搜索引擎准备的，用户通常是看不见它们的。

本章会带你创建一个基本的 HTML 页面，讨论一些最佳实践，并分别阐述上述三种主要成分。

① SVG 是 Scalable Vector Graphic（可缩放矢量图形）的简称。——译者注

注意：正如在前言中提到的，本书使用 HTML 泛指这门语言本身。如果需要突出 HTML 某一版本独有的特殊属性，则使用它们各自的名称。例如，“HTML5 引入了一些新的元素，并重新定义或删除了一些先前存在于 HTML 4 和 XHTML 1.0 中的元素。”更详细的说明参见前言中的“如何使用本书”。

1.1　基本 HTML 页面

来看一个基本的 HTML 页面，大概了解一下本章及后面的内容。图 1.1.1 显示了图 1.1.2 中 HTML 代码在桌面浏览器中显示出来的样子。我们会讲解图 1.1.2 中代码的基础知识。不过，即便你现在不能完全理解这些代码也不必担心，这只是让你初步了解 HTML，本书接下来的部分会介绍更多的 HTML 知识。

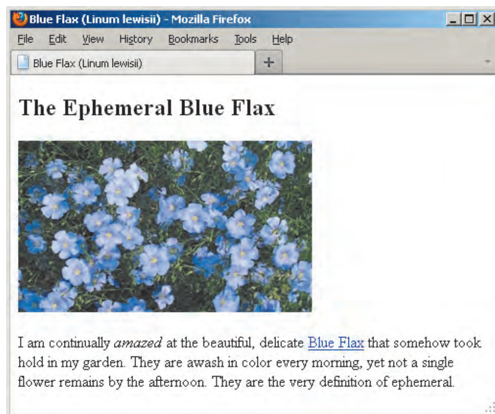


图 1.1.1　页面典型的默认呈现效果。虽然这是页面在 Firefox 中显示的效果，但该页面在其他浏览器中的效果也是相似的

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Blue Flax (Linum lewisii)</title>
</head>
<body>
    <article>
       <h1>The Ephemeral Blue Flax</h1>

       <img src="blueflax.jpg" width="300" height="175" alt="Blue Flax (Linum lewisii)"
/>

       <p>I am continually <em>amazed</em> at the beautiful, delicate <a href="http://
       → en.wikipedia.org/wiki/Linum_lewisii" rel="external" title="Learn more about
       Blue
       → Flax">Blue Flax</a> that somehow took hold in my garden. They are awash in color every
       → morning, yet not a single flower remains by the afternoon. They are the very definition
       → of ephemeral.</p>
    </article>
</body>
</html>
```

图 1.1.2　这是基本 HTML 页面的代码。我对 HTML 进行了突出显示，这样你就能把它们与页面中的文本内容区分开来。如图 1.1.1 中显示的那样，包围文本内容的 HTML 并没有在浏览器中显示出来。不过，你将了解到，这些标记是非常重要的，因为它们描述内容的含义。同时，请注意，每行之间都通过回车符分开了，但这并不是必需的，它们并不影响页面的呈现效果

你或许可以猜出某些代码的含义，特别是 body 部分的一些代码。首先看看 body 之前的那一部分代码。

在图 1.1.3 中，<body> 开始标记①以上的代码是为浏览器和搜索引擎准备的指导信息。每个网页都以 DOCTYPE 声明开头。该声明用以告诉浏览器这个页面的 HTML 版本。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Blue Flax (Linum lewisii) </title>
</head>
```

图 1.1.3　title 元素中的文本是 HTML 文档代码顶端部分中唯一用户可见的部分。其余有关页面的信息是为浏览器和搜索引擎准备的

应当始终使用 HTML5 的 DOCTYPE，即 <!DOCTYPE html>。代码不区分大小写，但 DOCTYPE 通常全部使用大写。无论如何，页面一定要包含 DOCTYPE（更多信息参见 3.1 节中的“改进后的 HTML5 DOCTYPE”）。

从 <!DOCTYPE html> 一直到 </head> 之间的部分对用户来说是不可见的，除了一处例外，即 <title> 和 </title> 之间的文字——Blue Flax (Linum lewisii)。这部分文字会作为标题显示在浏览器窗口顶端和标签页，如图 1.1.1 所示。此外，它们还通常是浏览器书签或收藏夹的默认名称，对搜索引擎来说也是很有价值的信息。第 3 章会解释页面顶部其他部分的作用。

网页的内容（即对用户可见的部分）位于 <body> 和 </body> 之间。最后，</html> 结束标记标志着页面的结束，如图 1.1.4 所示。

```
<!DOCTYPE html>
<html lang="en">
. . . [文档头部] . . .
<body>
    <article>
        <h1>The Ephemeral Blue Flax</h1>

        <img src="blueflax.jpg" width="300"
        → height="175" alt="Blue Flax (Linum
        → lewisii)" />

        <p>I am continually <em>amazed</em>
        → at the beautiful, delicate
        → <a href="http://en.wikipedia.org/
        → wiki/Linum_lewisii" rel="external"
        → title="Learn more about Blue Flax">
        → Blue Flax</a> that somehow took
        → hold in my garden. They are awash
        → in color every morning, yet not a
        → single flower remains by the
        → afternoon. They are the very
        → definition of ephemeral.</p>
    </article>
</body>
</html>
```

图 1.1.4　网页的内容位于 body 元素的开始标记和结束标记之间，文档以 </html> 结尾

代码的缩进与其是否为有效 HTML 毫无关系。它也不会影响内容在浏览器中的显示效果（第 4 章会讲到，pre 元素是一个例外）。不过，对嵌套在父元素里面的代码进行缩进是一种惯例，这样做会让你在阅读代码时更容易看出元素之间的层级关系。本章随后部分会讲到更多有关父元素和子元素的知识，以及更多有关浏览器默认显示的详细说明。

首先，让我们讨论一下编写语义化 HTML 是什么意思，以及为什么它是有效网站的基石。

① 开始标记指的是开启一个非空元素的一段代码（此处为 <body>），与结束标记对应（<body> 对应的结束标记为 </body>），参见 1.3 节。——译者注

1.2　语义化 HTML：有含义的标记

HTML 是一套精巧的系统，可以包含关于文档中内容的信息。这些信息称做标记，用以描述内容的含义（meaning），即语义（semantics）。基本 HTML 页面中已经出现了一些例子，如标记段落内容的 p 元素。

HTML 并不定义内容在浏览器应当如何显示，那是 CSS（Cascading Style Sheet，层叠样式表）的事。HTML5 比以往任何一个版本都更加强调这种区别，这是这门语言的核心。

既然如上所述，你或许想知道为什么基本 HTML 页面（如图 1.2.1 所示）中的一些文字比另一些文字要大一些，一些文字是粗体或斜体（如图 1.2.2 所示）。

```
...
<body>
    <article>
        <h1>The Ephemeral Blue Flax</h1>

        <img src="blueflax.jpg" width="300"
        → height="175" alt="Blue Flax (Linum
        → lewisii)" />

        <p>I am continually <em>amazed</em>
        → at the beautiful, delicate
        → <a href="http://en.wikipedia.org/
        → wiki/Linum_lewisii" rel="external"
        → title="Learn more about Blue Flax">
        → Blue Flax</a> that somehow took
        → hold in my garden. They ar awash
        → in color every morning, yet not a
        → single flower remains by the
        → afternoon. They are the very
        → definition of ephemeral.</p>

        <p><small>&copy; Blue Flax Society.
        → </small></p>
    </article>
</body>
</html>
```

图 1.2.1　基本页面的内容，结尾的地方又加了一段。HTML 元素并不描述内容如何显示，只说明内容的含义。内容的默认显示样式是由每个浏览器的内置样式表决定的，如图 1.2.2 所示

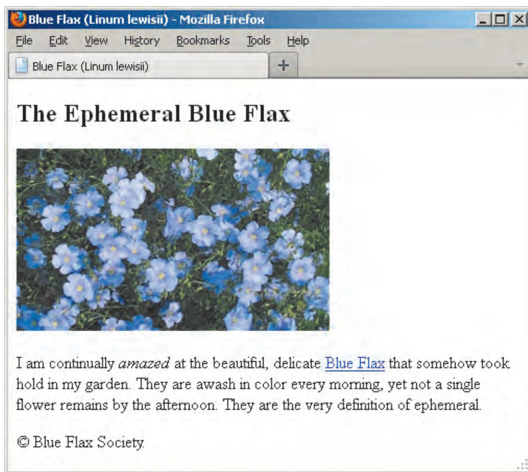


图 1.2.2　浏览器的默认样式表将标题（h1 ~ h6 元素）与普通文本区别开来，对 em 文本加上斜体样式，对链接加上颜色和下划线。此外，有的元素从单独的一行开始（如 h1 和 p），而其他一些元素显示在外围内容的里面（如 a 和 em）。这个例子显示了第二个段落（版权声明），可以清晰地看到每个段落占据单独的行。用你自己的样式表覆盖这些显示规则是很简单的

这是个好问题。其实，原因就是每个 Web 浏览器都有一个内置 CSS 文件（一张样式表），它决定了每个 HTML 元素的默认样式（你自己创建的 CSS 可以覆盖这些样式）。对于不同的浏览器，默认样式会稍有差异，但总体上是相当一致的。重要的是，HTML 所定义的内容的底层结构和含义是一致的。

1. 块级元素、行内元素以及 HTML5

容易看出，有些 HTML 元素（如 article、h1 和 p）各自显示为单独的一行，就像书里的各个段落一样，而另外一些元素（如 a 和 em）则与其他内容显示在同一行里，如图 1.2.2 所示。这也是浏览器的默认样式，而不是 HTML 元素自身的样式。这里需要多解释一下。在 HTML5 之前，大多数元素都可以划入块级（block-level，显示为单独的一行）或

行内（inline，可以与其他内容在一行并排显示）两种类别。HTML5 废弃了这些术语，因为这些术语把元素与表现关联起来了，而 HTML 并不负责表现。

通常，以前归为行内元素的在 HTML5 中称作*短语内容*（phasing content），即主要显示在一个段落之内的元素及其包含的文本。（第 4 章专门讲解短语内容，完整列表参见 http://dev.w3.org/html5/spec-author-view/content-models.html#phrasing-content-0。）

旧的块级元素也被归入了新的 HTML5 类别当中，这时强调的是它们的语义功能。这些元素大多用来标明内容的主要结构块和标题（关于给内容分块和标明标题级别的元素，参见第 3 章）。

尽管如此，浏览器不需要也不应该为这些元素改变默认显示规则。毕竟，你不想看到两个段落（p 元素）连到一起，或者强调的字句（em 元素）将句子打断，单独成行。

因此，通常标题、段落和结构化元素（如 article）显示在单独的行内，而短语内容则与外围内容显示在同一行。尽管 HTML5 不再使用块级、行内这些术语，但这样划分有助于理解它们的含义。由于在 HTML5 之前就成为 HTML 的常用词，因而这些术语常见于初级教程当中。本书也会偶尔使用这些术语，以说明元素在默认情况下是占据单独的行还是与其他内容共处一行。

后续章节将详细讲解 CSS，不过现在只需要知道，样式表与 HTML 一样是纯文本，因此你可以用编辑 HTML 的文本编辑器创建你自己的样式表。

2. HTML5 强调语义

HTML5 强调 HTML 的语义，把所有视觉样式都扔给了 CSS。但在 HTML 的早期版本中，并不总是这样。

万维网刚刚诞生那会儿，还没有为网页添加样式的恰当方法。在 CSS1 于 1996 年 12 月正式推出之前，HTML 就已经存在数年了。为了填补当时那段空白，HTML 包含了大量的呈现元素，这些元素用于给文本添加基本的样式，如加粗、斜体、改变字号等。

这些元素在当时发挥了作用，但随着 Web 开发最佳实践的发展，它们也理所当然地“失宠”了。这其中的核心观念就是，HTML 只负责描述内容的含义，而非表现。

HTML 的表现元素打破了这条最佳实践法则。因此，HTML4 不赞成使用它们，而推荐创建者使用 CSS 对文本和其他页面元素添加样式。

HTML5 则更进一步，它取消了一些表现元素，并重新定义了其他一些元素，使之仅具有语义功能，而非控制表现。

small 元素就是这样一个例子。起初，它用于让文字变得比常规文字小一些。然而，在 HTML5 中，small 代表条文细则，如通常用小字写的法律声明、免责条款等。当然，只要你愿意，可以使用 CSS 让它成为页面中最大的字，但这样做不会改变 small 内容的含义。

同时，HTML5 不存在与 small 相对的 big 元素。还有其他的一些例子，你会在本书接下来的章节了解到。

HTML5 也定义了新的元素，如 header、footer、nav、article、section 等。它们可以丰富内容的语义，后续章节会详述。

不过，无论是使用从 HTML 这门语言诞生起就存在的元素，还是使用 HTML5 的新元素，目标都应该是一样的：选择最能表明内容含义的元素，不必关心它们的表现。

3. 基本 HTML 页面的语义

熟悉了 HTML 的角色以后，下面就进一步看看对示例内容进行标记的过程。正如你所看到的，编写语义化 HTML 并不神奇。当

你熟悉了可供使用的元素以后，这基本上就是常识了。下面重温一下基本页面的 body 部分，看看最常用的 HTML 元素，如图 1.2.3 所示。

```
<body>
    <article>
        <h1>The Ephemeral Blue Flax</h1>

        <img src="blueflax.jpg" width="300"
        → height="175" alt="Blue Flax (Linum
        → lewisii)" />

        <p>I am continually <em>amazed</em>
        → at the beautiful, delicate
        → <a href="http://en.wikipedia.org/
        → wiki/Linum_lewisii" rel="external"
        → title="Learn more about Blue Flax">
        → Blue Flax</a> that somehow took
        → hold in my garden. They are awash
        → in color every morning, yet
        → not a single flower remains by
        → the afternoon. They are the very
        → definition of ephemeral.</p>
    </article>
</body>
```

图 1.2.3　基本页面的 body，它包含 article、h1、img、p、em 和 a 元素以描述内容的含义。所有的内容都嵌套在 article 中

所有的内容都包含在一个 article 元素里。简言之，article 定义了一段独立的内容。要包围基本页面内容，article 是个不错的选择，但不是每个页面都需要这样做。关于什么时候该用 article，参见第 3 章。

接下来是标题，参见图 1.2.4。HTML 提供了六个标题级别——h1 ~ h6。其中，h1 是最重要的一级。h2 是 h1 的次一级标题，h3 是 h2 的次一级标题，依次类推，就像用文字处理软件编辑一篇具有多级标题的文档一样。

```
<h1>The Ephemeral Blue Flax</h1>
```

图 1.2.4　标题是描述页面概貌的重要元素。它们使页面对屏幕阅读器用户来说更具可访问性，而搜索引擎用它们确定页面的重点

每个 HTML 页面都应该有一个 h1（或者多个 h1，这取决于你的内容）。因此，将标题标记为 h1 是比较直观的做法。关于 h1 ~ h6 标题元素，参见第 3 章。

接下来，我们插入一张图片，参见图 1.2.5。img 元素是显示图片的首选元素，因此也无需讨论用哪个元素合适。如果图片没有加载成功，或者页面是通过仅显示文本的浏览器查看的，就会显示 alt 属性中的文字。关于 img 元素，参见第 5 章。

```
<img src="blueflax.jpg" width="300"
→ height="175" alt="Blue Flax (Linum lewisii)"
→ />
```

图 1.2.5　用 img 在页面中插入图片很容易。如果图片未能显示，就会显示 alt 属性中的文字“Blue Flax (Linum lewisii)”

段落由 p 元素标记，如图 1.2.6 所示。同印刷材料中的段落一样，一个段落可以包含一个或多个句子。如果页面需要再加一个段落，只需要在第一个 p 元素之后再加一个 p 元素就可以了。

```
<p>I am continually <em>amazed</em> at
→ the beautiful, delicate <a href="http://
→ en.wikipedia.org/wiki/Linum_lewisii"
→ rel="external" title="Learn more about
→ Blue Flax">Blue Flax</a> that somehow
→ took hold in my garden. They are awash in
→ color every morning, yet not a single
→ flower remains by the afternoon. They are
→ the very definition of ephemeral.</p>
```

图 1.2.6　p 元素可以包含定义段落内短语语义的其他元素，em 和 a 是两个例子

在我们的段落里，有两个元素是用来定义少量文字的含义——em 和 a。这是 HTML5 提供的大量用于提升段落文本语义的短语内容元素的两个例子。它们与 p 会在第 4 章中讨论。

em 元素表示“强调”。在我们的示例页面中，它强调了对花的惊叹之情。记住，HTML 描述的是内容的含义，因此 em 代表的是语义上的强调，而非视觉上的（尽管通常会把 em 文本用斜体表示）。

最后，基本页面通过 a 元素定义了一个链接。链接可谓是所有 HTML 元素中最强大的元素，因为有它才得以形成万维网。万维网的定义就是，将一个页面与另一个页面或资源连接起来，或者将页面的一部分与另一部分连接起来（既可以是同一页面的两部分，又可以是不同页面的两部分）。在这个例子中，文字“Blue Flax”是一个指向维基百科某页面的链接，如图 1.2.7 所示。

```
<a href="http://en.wikipedia.org/wiki/Linum_
→lewisii" rel="external" title="Learn more
→about Blue Flax">Blue Flax</a>
```

图 1.2.7　a 元素定义了一个指向维基百科中关于 Blue Flax 的链接。rel 属性指出链接指向的是另一个网站，这也增加了语义。该属性是可选的，没有它链接也会正常工作。可选的 title 属性提供了有关所指页面的信息，它也增强了 a 元素的语义。当用户用鼠标滑过该链接，就会显示 title 属性里的内容

相当简单，对吧？当你对 HTML 元素了解更多之后，为内容选择正确的元素通常是相当简单的工作。偶尔，你会遇到一些内容，有一种以上的合理标记方式，这也没问题。有时候不能直接辨别对与错，当然大多数时候可以。

最后需要指出，HTML5 并未试着为每一种能想到的内容类型提供对应的元素，因为这样会使这门语言变得笨重。相反，HTML5 采取了一种务实的态度，其所定义的元素覆盖了绝大多数情况。

HTML 之美，部分在于人们能很容易地掌握其基础，从而构建一些页面，并在此基础上不断取得进步。因此，尽管有大约 100 种 HTML 元素，但不要被这个数字吓到。只有少量核心元素是你会经常用到的，而其余的则较少用到。你已经了解了一些常见元素，因此你已经有了个好的开始。

4. 为什么语义很重要

知道了语义化 HTML 的重要性，也看到了如何进行语义化以后，还需要知道它为什么如此重要。

下面是一些重要原因（当然还不全），其中的一些前面已经提到过了。

- 可以提升可访问性和互操作性（内容对于面向残障访问者的辅助技术是可用的，同时对于台式机、手机、平板电脑及其他设备上的浏览器都是可用的）。
- 改进了搜索引擎优化（SEO）。
- （通常）使代码更少，页面更快。
- 使维护代码和添加样式变得容易。

你可能对*可访问性*并不熟悉。它指的是让内容对所有用户可用，不论其能力如何（参见 www.w3.org/standards/webdesign/accessibility）。万维网的发明者 Tim Berners-Lee 曾说过一句著名的话：“万维网的力量在于其普适性。让包括残障人士在内的每个人都能访问万维网，是极为重要的一点。”

任何带有浏览器的设备都可以显示 HTML，因为它只是文本。然而，用户获取内容的方式可能并不相同。例如，视力正常的人可以直接查看内容，而视力受损的用户则需要放大页面，调大字号，或者使用屏幕阅读器（可以将内容朗读出来的软件，是辅助技术的一个例子）。有时，屏幕阅读器会将内容周围的 HTML 元素的类型读出来，让用

户了解上下文。例如，对于列表，在列表各个条目读出来之前，用户会首先被告知这里有一个列表。类似地，对于链接，用户会被告知这里有一个链接，方便其决定是否点击这个链接。

屏幕阅读器用户能够以多种方式浏览网页，例如通过键盘按键从一个标题跳到下一个标题。这样，他们可以先了解一个页面的关键主题有哪些，再去听他们感兴趣的内容，而不是必须把整个页面从头到尾听下来。

你看，对残障人士来说，好的语义产生了多么大的差别。

同时，SEO 的效果也会改善，也就是说网页在搜索引擎中的排名会靠前，因为搜索引擎对用特殊方式标记的内容会赋予更高的权重。例如，标题告诉搜索引擎爬虫页面的主要主题，帮助浏览器索引页面目录。

随着不断深入阅读本书，你会了解为什么好的语义能使代码更有效、更易于维护和添加样式。

1.3　标记：元素、属性和值

见识了一些 HTML 以后，我们来仔细看看标记的组成。

HTML 标记主要包括三种成分：元素（element）、属性（attribute）和值（value）。你已经在基本页面中见到了它们各自的实例。

1. 元素

元素就像描述网页不同部分的小标签一样：“这是一个标题，那是一个段落，而那一组链接是一个导航。”我们在前面已经讨论过一些元素。有的元素有一个或多个属性，属性进一步描述了元素的用途和内容（如果有的话）。

元素可以包含文本，可以包含其他元素，也可以是空的。一个非空元素由开始标记（start tag，元素的名称和属性，如果有的话放在尖括号中）、内容和结束标记（end tag，一个斜杠后跟元素的名称放在尖括号中）组成，如图 1.3.1 所示。

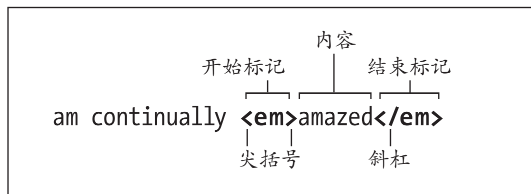


图 1.3.1　一个典型的 HTML 元素。开始标记和结束标记包着元素所描述的文字。在这个例子中，通过使用 em 元素，amazed 一词被强调了。习惯上，使用小写字母输入标记

空元素（empty element）看起来像开始标记和结束标记的结合，由左尖括号开头，然后是元素的名称和任何可能有的属性，然后是一个可选的空格和一个可选的斜杠，最后是必须有的右尖括号，如图 1.3.2 所示。

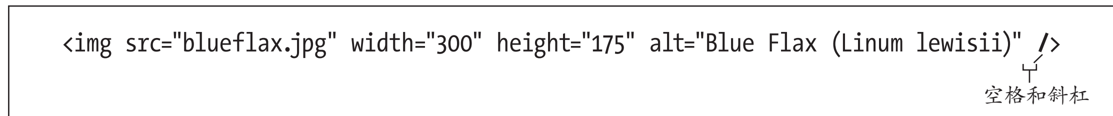


图 1.3.2　空元素并不包围任何文本内容（alt 属性中的文字是元素的一部分，并未被元素包围），就像这里显示的 img 元素。空元素只有一个标记，既作为元素开始，又作为元素结束。结尾处的空格和斜杠在 HTML5 中是可选的，但通常还是会写上它们。不过，元素最后面的 > 是必须有的

在 HTML5 中，空元素结尾处的空格和斜杠是可选的。XHTML 要求空元素结尾处有斜杠。应该说，我们这些以前用 XHTML 的人恐怕仍然倾向于在 HTML5 中继续这样做，而其他人则当然不用斜杠了。本书会在代码中包含斜杠，但如果选择忽略它们，页面也不会表现出任何不同。不管选择哪种方式，建议始终保持一致。

依照习惯，元素的名称都用小写字母，但 HTML5 对此也未做要求，用大写字母也是允许的。然而，现在很少有人用大写字母编写代码，因此，除非无法抗拒，否则不推荐这样做。使用大写字母是一种过时的做法。

2. 属性和值

属性包含的是有关文档内容的信息并非文档内容本身，如图 1.3.3 和图 1.3.4 所示。在 HTML5 中，属性值两边的引号是可选的，但习惯上还是会写上它们，因此建议始终这样做。同元素的名称一样，建议你用小写字母编写属性的名称。

本书会对大多数属性可接受的值进行详细说明，不过不妨先看看都有哪些类型的值。

有的属性可以接受任何值，有的则有限制。最常见的还是那些仅接受枚举的或预定义的值的属性。也就是说，必须从一个标准列表中选一个值，如图 1.3.5 所示。一定要用小写字母编写枚举的值。

```
      for 是 label 的一个属性
<label for="email">Email Address</label>
             for 属性的值
```

图 1.3.3　这是一个 label 元素（该元素用于将一个文字标签与一个表单域关联在一起），它有一对简单的属性和值。属性总是位于元素的开始标记里面，通常用一对引号包围属性的值

```
  href 是 a 的一个属性                               rel 也是 a 的一个属性
                        href 的值                          rel 的值
<a href="http://en.wikipedia.org/wiki/Linum_lewisii" rel="external"
→title="Learn more about Blue Flax">Blue Flax</a>
                   title 的值
  title 是 a 的一个属性
```

图 1.3.4　有的元素可以有一个或多个属性，每个属性都有各自的值，就像上面的 a 元素一样。属性的顺序并不重要。不同的属性值对之间都用空格隔开

```
<link rel="stylesheet" media="screen" href="blueflax.css" />
                               预定义的值
```

图 1.3.5　有的属性只接受特定的值。例如，link 元素里的 media 属性只能被设为 all、screen、print 等，你不能像对 title 属性那样任意输入一个值

很多属性的值需要填入一个数字，特别是那些描述大小和长度的值。数字值无需包含单位，只需输入数字本身。图片和视频的宽度和高度是有单位的，不过会默认它们的单位为像素。

有的属性引用其他文件，如 href 和 src。它们只能包含 URL（统一资源定位符，是万维网上文件的唯一地址）形式的值。关于 URL，参见 1.7 节。

3. 父元素与子元素

如果一个元素包含另一个元素，它就是被包含元素的父元素，被包含元素称为子元素。子元素中包含的任何元素都是外层的父元素的后代，如图 1.3.6 所示。实际上，可以创建网页的家谱，显示页面上各元素之间的层次关系并唯一地标识每个元素。

这种家谱式的基本结构是 HTML 代码的一个关键特性，它有助于在元素上添加样式（从第 7 章开始讨论）和应用 JavaScript 行为。

值得注意的是，当元素中包含其他元素时，每个元素都必须正确地嵌套，也就是子元素必须完全地包含在父元素中。使用结束标记的时候，它必须与最近的开始标记对应。换句话说，先开始元素 1，再开始元素 2，就要先结束元素 2，再结束元素 1，如图 1.3.7 所示。

```
<article>
   <h1>The Ephemeral Blue Flax</h1>
   <img src="blueflax.jpg"... />
   <p>... continually <em>amazed</em> ... delicate <a ...>Blue Flax</a> ...</p>
</article>
```

图 1.3.6　article 元素是 h1、img 和 p 元素的父元素。反过来，h1、img 和 p 元素是 article 元素的子元素（和后代）。p 元素是 em 和 a 元素的父元素。em 和 a 元素是 p 元素的子元素，也是 article 元素的后代（但不是子元素）。反过来，article 元素是它们的祖先

正确（没有重叠线）

```
<p>... continually <em>amazed</em> ...</p>
<p>... continually <em>amazed ...</p></em>
```

错误（标记对相互交叉）

图 1.3.7　元素必须正确地嵌套。如果先开始 p，再开始 em，就必须先结束 em，再结束 p

1.4　网页的文本内容

元素中包含的文本可能是网页上最基本的成分。如果你用过文字处理软件，那么你一定输入过文本。不过，HTML 页面中的文本有一些重要的差异。

首先，浏览器呈现 HTML 时，会把多个空格或制表符压缩成单个空格，并把回车和换行符转换成单个空格，或者将它们一起忽略，如图 1.4.1 和图 1.4.2 所示。

```
<p>I am continually <em>amazed</em> at the
→ beautiful,    delicate Blue Flax that
→ somehow took hold in my garden.

They are awash in       color every
→ morning, yet not a single flower
→ remains by the afternoon.



They are the very definition of
→ ephemeral.</p>
<p>&copy; Blue Flax Society.</p>
```

图 1.4.1　页面的文本内容（粗体部分）几乎就是标记以外的所有东西。在这个例子里面，注意每个句子都至少包含一个回车，有的词之间隔了好几个空格（为了说明对回车和空格的压缩）。另外，这里还包含一个表示版权符号的特殊字符引用（©），这样做可以确保无论以哪种编码方式保存这份文档，这个符号都会被正确地显示

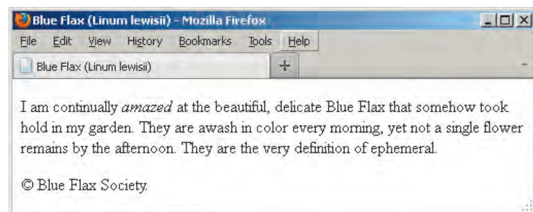


图 1.4.2　注意，在使用浏览器查看这份文档时，多余的回车和空格都被忽略了，字符引用被替换成了对应的符号（©）

其次，HTML 过去只能使用 ASCII 字符。ASCII 只包括英语字母、数字和少数几个常用的符号。重音字符（在很多西欧语言中很常见）和许多日常符号必须用特殊的字符引用来创建，如 é（表示 é）、©（表示 ©）等。完整列表见 www.elizabethcastro.com/html/extras/entities.html。

Unicode 大大减轻了特殊字符问题的负担。用 UTF-8 对页面进行编码（如基本页面那样，参见图 1.4.3），并用同样的编码保存 HTML 文件（参见 2.3 节）已成为一种标准做法。推荐你也这样做。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Blue Flax (Linum lewisii)</title>
</head>
<body>
...
</body>
</html>
```

图 1.4.3　直接在 head 开始标记后面指明文档的字符编码。charset 属性用以设置编码类型

由于 Unicode 是 ASCII 的超集（它包含 ASCII 中的所有字符，还包含许多其他的字符），因此用 Unicode 编码的文档与现有的浏览器和编辑器都兼容（特别旧的除外）。不理解 Unicode 的浏览器会正确地解释文档的 ASCII 部分，而理解 Unicode 的浏览器还会显示非 ASCII 部分。即便如此，有时还是会使用字符引用，如版权符号（因为 © 既好记又好输入），如图 1.4.1 所示。

1.5　链接、图像和其他非文本内容

显然，万维网充满生机的部分原因是页面之间的链接，以及图像、视频、音乐、动画等。外部文件（如视频）实际上并没有放在 HTML 文件中，而是单独保存的，只是简单地在页面中引用了这些文件，如图 1.5.1 所示。由于引用只是文本，因此不会破坏 HTML 文件的普遍可访问性。

```
...
<article>
    <h1>The Ephemeral Blue Flax</h1>

    <img src="blueflax.jpg" width="300"
    ➝ height="175" alt="Blue Flax (Linum
    ➝ lewisii)" />

    <p>I am continually <em>amazed</em> at
    ➝ the beautiful, delicate <a href=
    ➝ "http://en.wikipedia.org/wiki/
    Linum_
    ➝ lewisii" rel="external" title="Learn
    ➝ more about the Blue Flax">Blue Flax
    ➝ </a> that somehow took hold in my
    ➝ garden. They are awash in color every
    ➝ morning, yet not a single flower
    ➝ remains by the afternoon. They are the
    ➝ very definition of ephemeral.</p>
</article>
...
```

图 1.5.1　在我们的基本 HTML 文档中，有一个对图像文件 blueflax.jpg 的引用，浏览器在加载页面其他部分的同时，会请求、加载和显示这个图片。该页还包括一个指向关于 Blue Flax 的页面的链接

浏览器可以毫无困难地处理链接和图像（仅支持文本的浏览器除外），如图 1.5.2 所示。然而，它们无法处理其他任何文件类型。如果引用了访问者的浏览器不能理解的文件，浏览器通常会试着寻找能打开这种文件的插件或辅助应用程序（位于访问者计算机上的某个程序）。

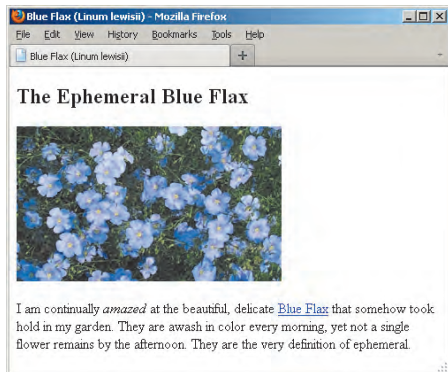


图 1.5.2　从网页引用图像和其他非文本内容，浏览器会将这些内容与文本一起显示

还可以向浏览器提供额外的信息，告诉它呈现内容需要用到的插件，以及如果访问者的计算机上没有查看特定文件的插件，应该如何下载这些插件。

下载和安装插件会打断用户访问网站的体验，让人感觉完成这些事情需要花很长时间。由于插件并不是浏览器自带的部分，它们还会引入性能问题。

例如，Flash 是多年来使用最为广泛的插件。你肯定经历过观看 Flash 视频的过程中计算机变慢甚至浏览器崩溃的情况。

HTML5 试图减轻这些问题，其做法是通过 audio 和 video 元素在浏览器中引入原生的媒体播放器。不幸的是，不同的浏览器厂商对支持的媒体格式存在争议，因此还无法完全抛弃插件。不过，这毕竟是个开始。

本书将在第 5 章讨论图像，在第 17 章讨论插件、HTML5 的媒体元素等内容。

1.6　文件名

同其他文本文档一样，网页也有一个文件名，用以向创建者、访问者及访问者的 Web 浏览器标识该文件。对网页文件命名时要记住几点，这些要点有助于对文件进行组织，使访问者更容易找到并访问你的页面，确保他们的浏览器能正确地处理页面，以及增强 SEO，请见图 1.6.1 和图 1.6.2。

1. 使用小写字母的文件名

为网页选择的文件名决定了访问者在访问你的页面时需要输入什么，因此如果文件名只用小写字母的话，访问者就能避免无意的输入错误（同时避免为此而头疼）。这样做对创建页面之间的链接也有很大的帮助。如果所有的文件名都用小写字母，你就少了一件需要操心的事情。

2. 用短横线分隔单词

不要在文件名中使用空格分隔单词。应该使用短横线，例如 company-history.html 和 my-favorite-movies.html。有的网站使用下划线（“_”），但这不是推荐的做法，因为短横线是搜索引擎更倾向于接受的方式。

3. 使用正确的扩展名

浏览器主要通过查看文件的扩展名得知需要读取的文本文档是一个网页。尽管可以使用 .htm 作为网页的扩展名，但通常使用的是 .html，因此推荐使用 .html。如果页面使用其他的扩展名（如 .txt），浏览器会将其当做文本处理，于是会把你的代码直接呈现给访问者。

> **提示** Mac OS 和 Windows 并不总是显示文档的实际扩展名。如需查看扩展名，请更改文件夹选项。

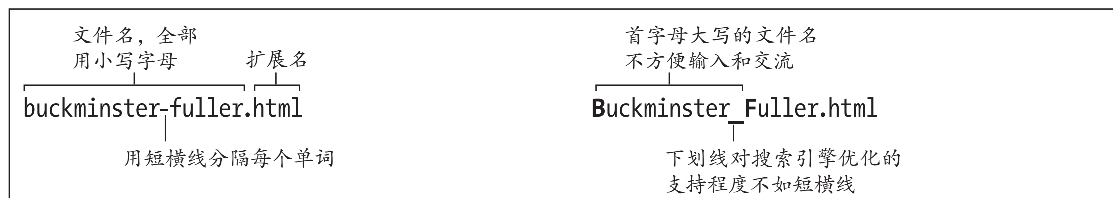


图 1.6.1 记住，文件名全部使用小写字母，用短横线分隔单词，用 .html 作为扩展名。混合使用大小写字母会增加访问者输入正确地址以及找到页面的难度

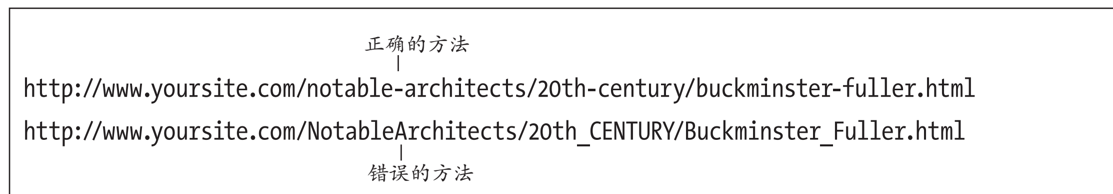


图 1.6.2 目录和文件夹的名称也应全部用小写字母，关键是保持一致。如果不使用大写字母，访问者和创建者就不必浪费时间思考：“这个字母是大写的 B 还是小写的？”

1.7 URL

URL（Uniform Resource Locator，统一资源定位符）是地址的别名。它包含关于文件存储位置和浏览器应如何处理它的信息。互联网上的每个文件都有唯一的 URL。

URL 的第一个部分称为模式（scheme）。模式告诉浏览器如何处理需要打开的文件。最常见的模式是 HTTP（Hypertext Transfer Protocol，超文本传输协议），用于访问网页，如图 1.7.1 所示。

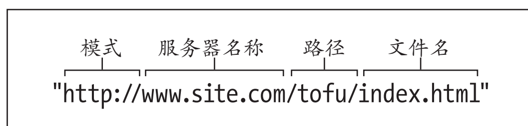


图 1.7.1 基本的 URL 包含模式、服务器名称、路径和文件名

URL 的第二个部分是文件所在服务器的名称，紧接着是到达这个文件的路径，以及文件自身的名称。有时 URL 不以文件名结尾，而以一个路径结尾（可以包含一个结尾的斜杠，也可以不包含），如图 1.7.2 所示。在这

种情况下，URL 指的是路径中最后一个目录中的默认文件（通常为 index.html）。



图 1.7.2　以一个斜杠而非文件名结尾的 URL 指向最后一个目录（在这个例子中是 tofu 目录）中的默认文件。最常见的默认文件名是 index.html。因此，这个 URL 与上一个例子中的 URL 指向的是同一个页面

其他常见的模式有 https（用于安全网页）、ftp（File Transfer Protocol，文件传输协议，用于下载文件，参见图 1.7.3）、mailto（用于发电子邮件，参见图 1.7.4）、file（用于访问本地硬盘或本地文件共享网络上的文件，这种模式很少有机会用到，参见图 1.7.5）。

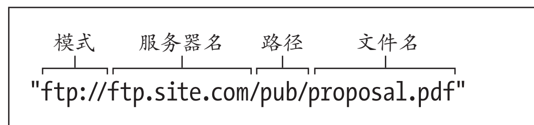


图 1.7.3　当用户点击这个 URL 时，浏览器会开始按 FTP 模式传输文件 proposal.pdf

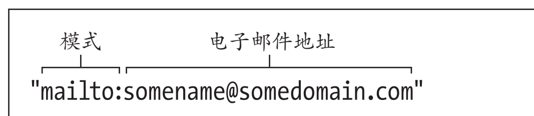


图 1.7.4　用于电子邮件地址的 URL 包括 mailto 模式，紧接着是一个冒号（没有斜杠），然后是电子邮件地址本身

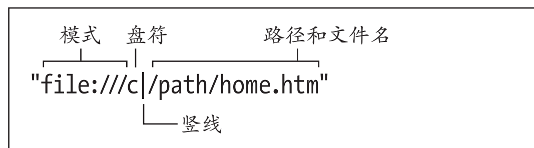


图 1.7.5　使用 file 模式引用本地 Windows 机器上的文件。对于 Macintosh，应使用 file:///Harddisk/path/filename，无需竖线。有时在 Windows 上也不需要竖线

模式后面通常紧跟一个冒号和两个斜杠。mailto 和 news 是例外，它们后面只有一个冒号。

注意 file 模式后面跟着一个冒号和三个斜杠。这是因为第二个和第三个斜杠之间代表的主机假定为本地计算机。应始终用小写字母输入模式。

在上述模式中，最常用到的是 http 和 mailto，其他模式则只有在特殊情况下才会用到。

1. 绝对 URL

URL 可以是绝对的，也可以是相对的。绝对 URL（absolute URL）显示文件的完整路径，包括模式、服务器名称、完整路径和文件名本身，如图 1.7.6 所示。绝对 URL 就像是完整的通信地址，包括国家、州、城市、邮政编码、街道和姓名。无论邮件来自哪里，邮局都能找到收件人。对于 URL，这意味着绝对 URL 本身与被引用文件的实际位置无关，无论是在哪个服务器上的网页中，某一文件的绝对 URL 都是完全一样的。



图 1.7.6　包含 URL 的文档（在这个例子中是 you-are-here.html）是相对 URL 的参照点。换句话说，相对 URL 是相对于这个文件在服务器上的位置。绝对 URL 与其所在的位置无关，因为它们总是包含资源的完整 URL

引用别人的服务器上的文件时，总是应该使用绝对 URL。对于 FTP 站点以及几乎所有不使用 HTTP 协议的 URL，都应该使用绝对 URL。

表 1.7.1 描述了访问 you-are-here.html 中各个文件的方式，作为对相对 URL 和绝对 URL 的比较。表中包括这些文件与 you-are-here.html 在同一站点（site.com）和在另一站点（remote.com）的情况。

表 1.7.1 绝对 URL 与相对 URL

文 件 名	绝对 URL（可以在任何地方使用）	相对 URL（只能在 you-are-here.html 中使用）
index.html	http://www.site.com/about/index.html	index.html
data.html	http://www.site.com/about/info/data.html	/info/data.html
image.png	http://www.site.com/img/image.png	../img/image.png
news.html	http://www.remote.com/press/news.html	无，请使用绝对 URL
index.html	http://www.remote.com/sign-up/index.html	无，请使用绝对 URL

2. 相对 URL

当我告诉你我邻居家的位置时，我一般不会说完整地址，而是说：“她家在右边第三个门。”这就是相对地址，它指出的位置是以信息提供者的位置为参照的。如果在别的城市按照同样的信息找我的邻居，你永远也找不到。

同样，相对 URL 以包含 URL 本身的文件的位置为参照点，描述目标文件的位置。因此，相对 URL 可以表达像“指向本页面同一目录下的 xyz 页面”这样的意思。

如果目标文件与当前页面（也就是包含 URL 的页面）在同一个目录中，那么这个文件的相对 URL 就只有文件名和扩展名，如图 1.7.7 所示。如果目标文件在当前目录的子目录中，那么这个文件的相对 URL 就是子目录名，接着是一个斜杠，然后是文件名和扩展名，如图 1.7.8 所示。

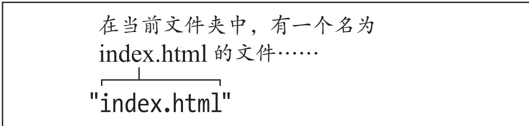


图 1.7.7 相对 URL 指向同一目录下的文件（参见图 1.7.6）。URL 中只需要文件名和扩展名，不必在前面加上 http://www.site.com/about/（这两个文件所在的目录）

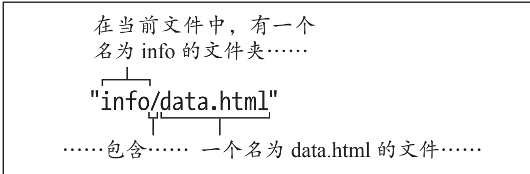


图 1.7.8 要引用当前文件夹的子文件夹中的文件（在这个例子中是 data.html）（参见图 1.7.6），应在文件名之前加上子文件夹名称和一个斜杠

如果要引用文件层次结构中更高层目录中的文件，那么应该使用两个句点和一个斜杠，如图 1.7.9 所示。可以组合和重复使用两个句点和一个斜杠，引用当前文件所在服务器或硬盘上的任何文件。

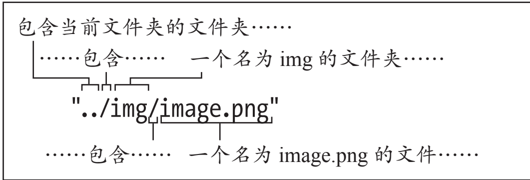


图 1.7.9 在图 1.7.6 中可以看到，这个文件位于与网站根目录下的当前文件夹（about）同属一层的文件夹（img）中。在这种情况下，使用两个句点和一个斜杠上升一级，然后指出子目录，再跟一个斜杠，最后是文件名。（实践中，可以为 image.png 选择一个更具描述性的文件名，这里的名称仅作为示例。）

不过，如果文件位于 Web 服务器上，应该避免使用像 ../../img/family/vacation.jpg 这样显得较为笨拙的文件路径，应先直接跳到网站的根目录再顺着向下找到目标文件。可以在最开始使用一个斜杠，这样本例中的根相对 URL 就是 /img/family/vacation.jpg（假定 img 文件夹位于网站的根文件夹，这也是惯常的用法）。需要强调的是，这种做法只能用于 Web 服务器上，例如在网站托管服务供应商的 Web 服务器上，或者在本地计算机运行的 Web 服务器上（Apache 是最流行的 Web 服务器）。

如果不是在服务器本地开发网站，通常应使用相对 URL（当然，除非是指向其他服务器上文件的情况）。这样做会使将页面从本地系统传到服务器变得容易。只要每个文件的相对位置保持不变，就不必修改任何路径，链接依然有效。

1.8　要点回顾

HTML 基础以及某些关键最佳实践为构建有效网站打下了基础。下面我们来回顾一下本章讲述的要点。

- 一个网页主要由三种成分构成：文本内容、对其他文件的引用和标记。
- HTML 标记由元素、属性和值构成。
- 通常全部使用小写字母编写 HTML（DOCTYPE 是一个例外），用引号包住属性值，用一个空格和斜杠（ /）结束空元素。
- 始终用下面的 DOCTYPE 声明开始 HTML 文档。

```
<!DOCTYPE html>
```

- 页面内容都在 body 元素中。主要为浏览器和搜索引擎准备的指令位于 body 元素之前，在 head 元素中。
- 用语义化 HTML 标记内容，不关心它在浏览器中显示的样式。
- 语义化 HTML 提升网站的可访问性，让网站更有效率，并让网站维护和添加样式变得更容易。
- CSS 控制 HTML 内容的表现。
- 每个浏览器自带的样式表规定 HTML 的默认表现样式。可以使用自己写的 CSS 覆盖这些规则。
- 全部使用小写字母创建文件和文件夹的名称，用短横线而非空格或下划线对单词进行分隔。

接下来你会了解如何处理网页文件。

第 2 章

处理网页文件

本章内容

- ❑ 规划网站
- ❑ 创建新的网页
- ❑ 保存网页
- ❑ 指定默认页面或主页
- ❑ 编辑网页
- ❑ 对文件进行组织
- ❑ 在浏览器中查看网页
- ❑ 借鉴他人灵感

开始编写 HTML 元素和属性代码之前，有必要了解如何创建使用这些代码的文件。在本章中，你将了解到如何创建、编辑和保存网页文件，我们还会涉及一些设计和组织方面的注意事项。

如果你已知道如何创建文件，没有耐心读完本章，可以直接跳到讲解 HTML 代码本身的第 3 章。

2.1　规划网站

尽管可以直接开始编辑网页，但最好还是先对网站进行思考和规划，如图 2.1.1 所示。这样，你就不会迷失方向，而且将来的重组工作也会减少。

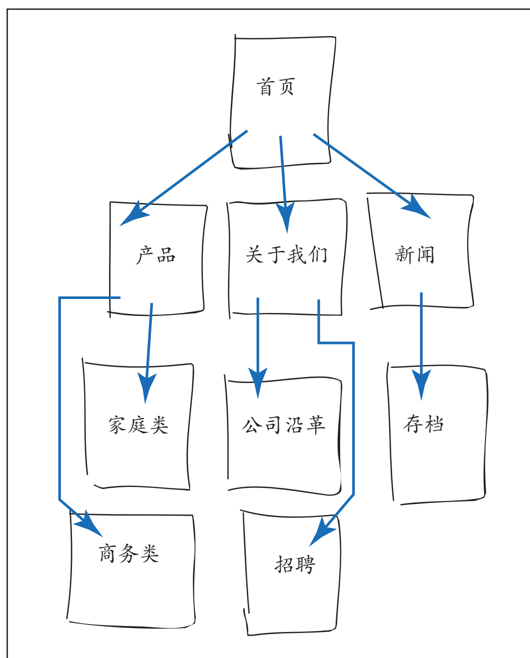


图 2.1.1　画出网站结构草图，考虑它可能包含的内容，这有助于创建者决定它需要何种结构

规划网站的步骤

- ❑ 确定为什么要创建这个站点，需要展示的内容是什么？
- ❑ 考虑站点的访问者。应该如何调整内容使之吸引这些访问者？

- ❑ 需要多少个页面？你希望网站是怎样的结构？你是希望访问者以某种特定的次序浏览网站，还是希望访问者可以自由地探索？
- ❑ 在纸上画出站点结构的草图。
- ❑ 为页面、图像和其他外部文件设计一个简单且一致的命名规则（参见 1.6 节）。

提 示　不要在站点规划阶段花费过多时间。到了适当的程度，就可以开始编写内容和代码了。

提 示　如果你对万维网还不太熟悉，先上网逛逛，从而了解可能的网站形式。可以从竞争对手的网站开始入手。

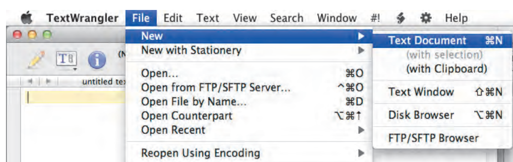
提 示　在纸上画出网站组织的文件夹结构（如图 2.1.1 所示）是一项常见的工作，但并非必需的。参见 2.6 节。

提 示　Erin Kissane 的文章“A Checklist for Content Work”（www.alistapart.com/articles/a-checklist-for-content-work/）的一部分讲解了如何开始制作网站内容。这也是她关于内容战略的书的一部分。

提 示　如果你不是设计师，或者只是一位设计新手，正在寻找关于如何创建有吸引力的、有效的站点的指导，Jason Beaird 的《完美网页的视觉设计法则》（*The Principles of Beautiful Web Design*，SitePoint，2010）或许会引起你的兴趣。

2.2　创建新的网页

创建网站并不需要特殊的工具。你可以使用任何文本编辑器，甚至是 Windows 自带的记事本 Notepad，如图 2.2.1 和图 2.2.3 所示，或者 OS X 上的免费软件 TextWrangler（www.barebones.com/products/textwrangler），如图 2.2.1 和图 2.2.2 所示。（Mac 也自带一款编辑器，名为 TextEdit，但它在 OS X 某些版本中存在漏洞，使之无法正确处理 HTML 文件。）



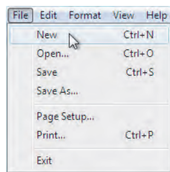


图 2.2.1　打开文本编辑器。在编辑器中出现的空白文档中输入 HTML，或者先选择 File → New。实际的菜单选项可能略有差异。如果使用 TextWrangler（Mac），就是 File → New → Text Document（文件→新建→文本文档），如上半部分所示。下半部分显示的是 Notepad（Windows）的选项

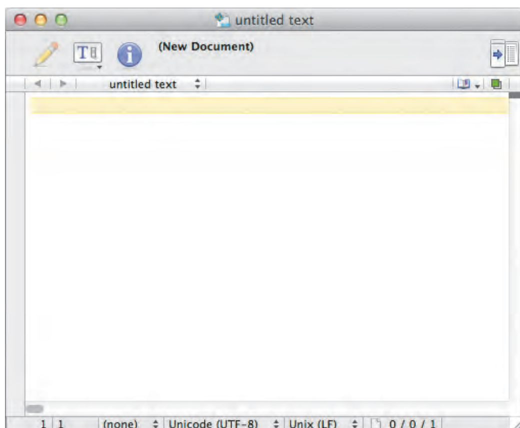


图 2.2.2　在 Mac 上，可以使用 TextWrangler 编写网页的 HTML 代码。下文提示部分列出了一些功能更强的编写代码的 Mac 编辑器

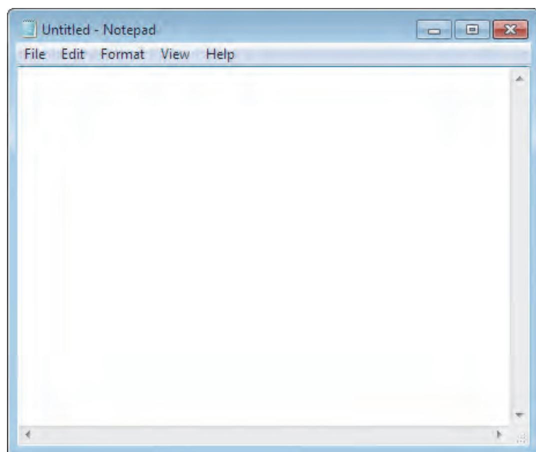


图 2.2.3　这是记事本，Windows 用户用以创建 HTML 页面的最基本的程序。也可以使用其他的编辑器（参见提示）

创建新网页的步骤

(1) 打开文本编辑器。

(2) 选择 File → New（文件→新建）创建一个新的空白文档，如图 2.2.1 所示。

(3) 按照本书其余部分的解释（从第 3 章开始）创建 HTML 内容。

(4) 一定要按照 2.3 节中的说明保存文件。

提 示　在 OS X 和 Windows 上都有很多专门编写 HTML（和 CSS）代码的文本编辑器。它们的代码提示和代码补全功能让你可以更准确、更快速地编写代码；它们会将代码突出显示，方便区分 HTML 元素和其中的文本内容；它们还有其他有用的功能。记事本没有这些功能。有不少免费的 HTML 编辑器，不过一些收费的编辑器也值得购买，通常还可以在购买之前进行试用

提 示　OS X 的流行编辑器包括 BBEdit（www.barebones.com/products/bbedit/）、Coda（www.panic.com/coda/）、Espresso（http://macrabbit.com/espresso/）、Sublime Text（www.sublimetext.com）以及 TextMate（http://macromates.com）。（通常可以认为 TextWrangler 是 BBEdit Lite 的简化版。）其中 TextMate 是最流行的，尽管竞争对手也在蚕食其用户群。Sublime Text 也有 Windows 版本。Windows 下的编辑器还有很多，如 E Text Editor（www.e-texteditor.com）、Notepad++（http://notepad-plus-plus.org），等等。在网上搜索"HTML 编辑器"可以找到更多的编辑器。

提 示　使用上面列出的某种编辑器创建新网页的方法也是类似的。要编辑已经存在的页面，只需在文本编辑器中选择 File → Open（文件 → 打开）并打开该文件（参见 2.4 节）。可以参照本书后面的章节为页面添加自己的 HTML 和 CSS。

提 示　请不要使用 Microsoft Word 这样的文字处理软件编写 HTML 页面。它们会在文件里添加一些无用的或无效的代码。

2.3　保存网页

用文本编辑器创建的网页需要在多种平台上的多种浏览器查看。为了让所有这些程序都能访问网页，网页需要保存为通用的"纯文本"格式，不包含文字处理软件可能应用的任何专用格式化信息。

为了让浏览器（和服务器）能识别网页并知道解释其中包含的标记，网页文件应在文件名中使用 .html 或 .htm 作为扩展名。这样做也可以将网页文件与不是网页的普通文本文件区分开来。上述两种扩展名都可以用，但通常还是使用 .html，因此推荐读者使用 .html 作为文件扩展名。

有了 .html 扩展名，网页的图标会显示为系统默认浏览器的图标，而不是用来编写这个文件的编辑器的图标，如图 2.3.1 所示。双击网页文件会在浏览器中打开它，而不是在文本编辑器中。这对于在浏览器中测试页面来说很方便，但却为编辑网页增加了一个额外的步骤（参见 2.5 节）。



图 2.3.1　Excel 工作簿文件的扩展名为 .xlsx，用 Excel 的图标进行标识（上图）。如果双击它，就会在 Excel 中显示它。对于网页文件，无论用哪种文本编辑器创建，它们的扩展名都是 .html 或 .htm，并采用系统默认浏览器（这里是 Firefox）的图标进行标识。双击它会在默认浏览器（而不是文本编辑器）中显示

总之，保存文件时，需要将文件保存为纯文本格式，并使用 .html 或 .htm 作为扩展名。

保存网页的步骤

(1) 创建网页之后，在文本编辑器中选择 File → Save As（文件 → 另存为），如图 2.3.2 所示。

(2) 在随后弹出的对话框中，选择纯文本或文本文档（或别的叫法）作为文件格式。

(3) 为文档添加 .html 或 .htm 的扩展名（这一点非常重要）。

(4) 选择要保存网页的文件夹。

(5) 点击 Save（保存），如图 2.3.3 和图 2.3.4 所示。

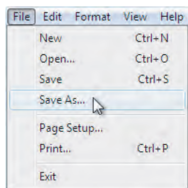


图 2.3.2　在文本编辑器中选择 File → Save As

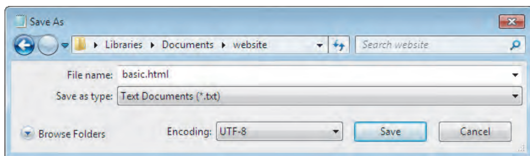


图 2.3.3　在记事本中，为文件名添加 .html 或 .htm 的扩展名，在 Save as type（保存类型）下拉菜单中选择文本文档，确保 Encoding（编码）选择的是 UTF-8（参见最后一项提示），点击 Save。在其他文本编辑器中，选项可能并不相同（不过是相似的）

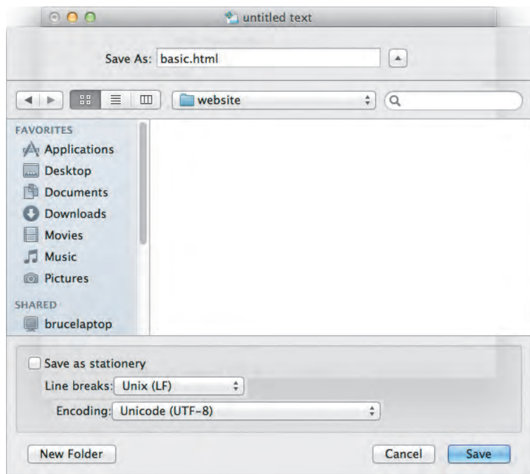


图 2.3.4　在 TextWrangler 中，为文件起名，选择保存位置。TextWrangler 默认以 UTF-8 进行编码（一般情况下，这就是应该选用的编码），不过你也可以从 Encoding 下拉菜单中选择其他编码。点击 Save 保存文件

提 示　使用 .html 还是 .htm 并无区别，但推荐使用 .html，因为它更常见。无论你用哪种扩展名，请保持一致，因为使用相同的扩展名有助于后续记住 URL。

提 示　即使已经指定了 .html 或 .htm 的扩展名，Windows 上的某些文本编辑器也会在文件名末尾加上默认扩展名。（注意大多数专门编

辑 HTML 页面的编辑器并不存在这个问题。）这样文件名就变成了 webpage.html.txt，这样的文件无法在浏览器中查看。Windows 通常会隐藏扩展名，也导致这个问题变得隐蔽，它尤其容易困扰新手。有两个解决办法：一个是在首次保存文件时将文件名包围在双引号中，这样能防止添加额外的扩展名；另一个是让 Windows 显示文件扩展名，方法如图 2.3.5 所示，从而可以看见程序自动添加的扩展名并将其删除。

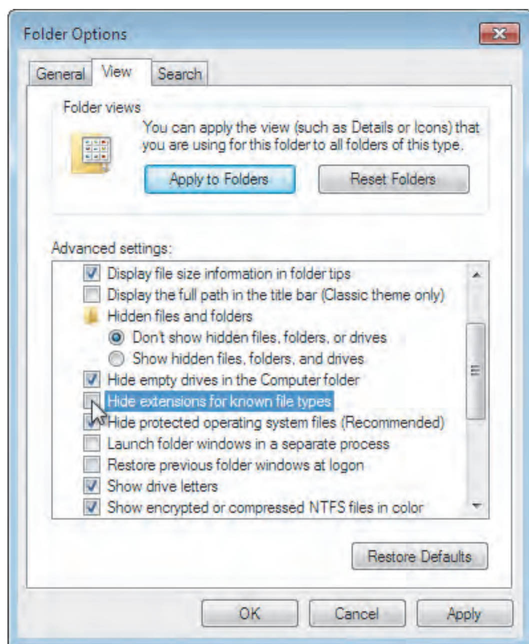


图 2.3.5　在 Windows 资源管理器中，选择 Organize → Folder and search options（组织 → 文件夹和搜索选项），或者 Tools → Folder Options（工具 → 文件夹选项）（取决于你使用的 Windows 版本），显示图中的对话框。点击 View（查看）选项卡，向下滚动，直到看到 Hide extensions for known file types（隐藏已知文件类型的扩展名）。如果想在桌面上看到文件的扩展名（如 .html），要确保这个选项是未选中的

提 示　选择将文件保存为纯文本格式时，文件会以系统默认字符编码保存。如果需要创建其他编码的网页（或者是因为网页里包含一些特殊符号或其他语言的文字），需要使用可以选择编码类型的文本编辑器。通常，UTF-8 是最好的选择。如果编辑器中可选择的文件编码类型包括“UTF-8，不含 BOM”或类似的选项，请选择该项。否则，就选择 UTF-8，如图 2.3.6 所示。有时，编辑器的 UTF-8 模式并不包含 BOM，但编码类型选择菜单中并未显式指出这一点。（BOM 的含义参见 http://en.wikipedia.org/wiki/Byte_order_mark。做好看不懂的准备吧！）

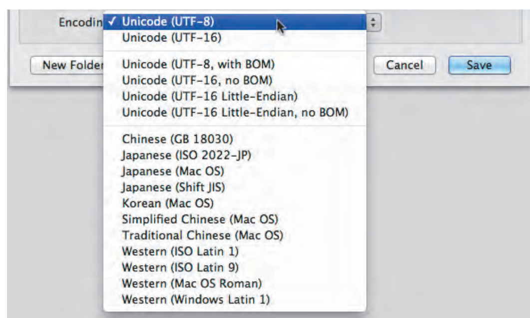


图 2.3.6　许多文本编辑器允许为文件选择编码，从而可以在同一个文档中保存不同语言中的符号和字符。大多数情况下，推荐选择 UTF-8。选择“UTF-8，不含 BOM”选项（如果有的话）。否则，直接选择 UTF-8。有的编辑器（如图中的 TextWrangler）默认选择该项

2.4　指定默认页面或主页

大多数服务器都会根据文件名识别每个文件夹中的默认页面。大多数情况下，系统会将 index.html 识别为默认页面，如图 2.4.1。如果没有 index.html，就会继续寻找 index.htm、default.htm 等文件名。如果访问者

输入带目录的 URL，但没有指定文件名，那么就会打开默认页面，如图 2.4.2 所示。



图 2.4.1　将文件保存为 index.html，从而指定这个文件是目录下的默认页面



图 2.4.2　当访问者输入目录的路径而没有指定文件名时，就会打开使用默认名称的页面。在这个例子中，输入的 URL 是 http://bruceontheloose.com/htmlcss/examples/antoni-gaudi/。如果输入 http://bruceontheloose.com/htmlcss/examples/antoni-gaudi/index.html，会显示相同的页面

在网站最顶层目录（通常称为根目录）中创建的默认页面（通常为 index.html）是网站的主页。当访问者只输入域名而没有指定路径信息（如 www.*yourdomain*.com）时，就会显示这个页面。

类似地，可以为网站的任何一个目录（甚至每一个目录）创建默认页面。例如，网站中 /products/ 或 /about-us/ 目录的着陆页（即首页）也应该是 index.html，只不过它们放在各自的目录里（目录就像是你自己计算机的文件夹）。网站的访问者通常通过主页或每页都有的主导航访问这些页面。

指定网站主页或含目录的着陆页的步骤

在目标文件夹中，将文件保存为 index.html（参见 2.3 节）。（当你将文件上传到网站的服务器以后，如果 index.html 无法起到默认页面的作用，请向 Web 主机提供商咨询。关于将网页上传到服务器，参见第 21 章。）

提 示　如果目录中没有默认页面，有的服务器就会将目录文件列表显示出来（你可能并不希望向访问者暴露这些内容）。为了防止这种情况的发生，应该在网站每一个包含 HTML 页面的目录下创建一个默认页面。另一种办法是修改服务器的配置，将文件列表隐藏起来（如果它们已被隐藏，也可以将它们显示出来）。对于包含图像、媒体文件、样式表以及 JavaScript 等资源的目录，建议将文件列表隐藏。如果不知道如何修改服务器配置，请向 Web 主机提供商咨询。

2.5　编辑网页

因为网页在大多数情况下是通过浏览器查看的，所以在桌面上双击网页文件，会启动默认浏览器并显示它们。如果想编辑网页文件，需要在文本编辑器中手工打开它。

编辑网页的步骤

(1) 打开文本编辑器。

(2) 选择 File → Open。

(3) 找到包含目标文件的目录。

(4) 如果没有看到目标文件，选择 All Files（全部文件）选项（或类似的叫法），如图 2.5.1 和图 2.5.2 所示。在不同的程序或平台，这个选项的叫法和位置都有细微的差异，如图 2.5.3 所示。

(5) 点击 Open，就可以开始编辑文件了。

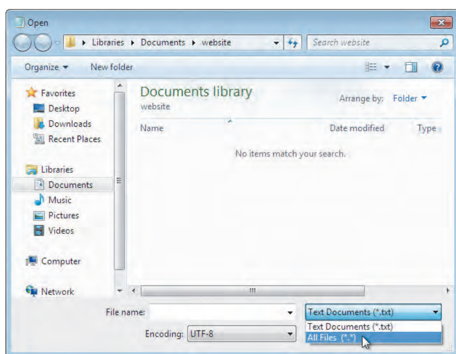


图 2.5.1　在 Windows 的一些文本编辑器（如记事本）中，不会自动看到 HTML 文件。选择 All Files 可以查看所有文件

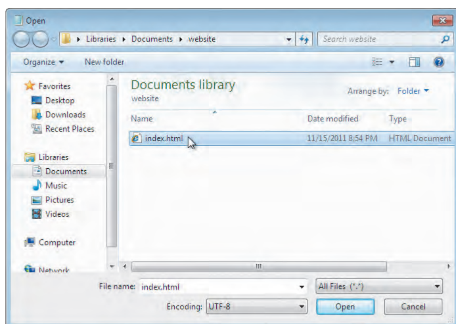


图 2.5.2　显示所有文件以后选中要编辑的 HTML 文件，点击 Open

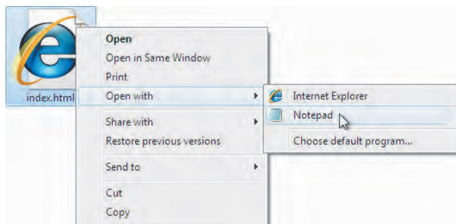


图 2.5.3　在 Windows 中，还可以右击文档的图标，在弹出菜单中选择 Edit 或 Open。在 Mac 上，右击图标，在弹出菜单中选择 Open With，再选择要使用的文本编辑器

提 示　通常，要保存对已有文档进行的修改，只需要选择 File → Save（文件 → 保存）即可，不必担心 2.3 节中提到的格式问题。

2.6　组织文件

在文件数量变得很大之前，最好考虑好将它们放在什么地方。通常（但非必须）为网站的主要区块创建单独的文件夹，使相关的 HTML 页面放在一起。

组织文件的步骤

(1) 创建一个主文件夹或目录存放网站上所有可用的资料。在 Mac 上，在 Finder 中选择 File → New Folder（文件 → 新文件夹），如图 2.6.1 所示。在 Windows 中，右击桌面（或你选择的文件夹），选择 New → Folder（新建 → 文件夹），如图 2.6.2 所示。然后为文件夹命名。

(2) 根据网站的组织结构创建子文件夹，如图 2.6.1 和图 2.6.3 所示。例如，可以考虑为网站的每个部分创建单独的文件夹，并根据需要在其中创建单独的子文件夹。

(3) 为网站的图片创建顶层文件夹是惯例，还可以根据网站结构划分或其他标准创建子文件夹，方便对图片进行组织。另一种方法就是创建名为 Assets 的顶层文件夹，将图片同其他资源（如视频、样式表等）的文件夹都放在其中。（我们将从第 7 章开始学习样式表。）

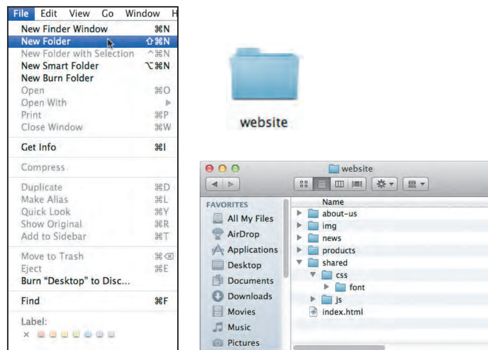


图 2.6.1　在 Mac 上，选择 New Folder，再为文件夹命名。为网站每个不同的部分创建独立的文件夹

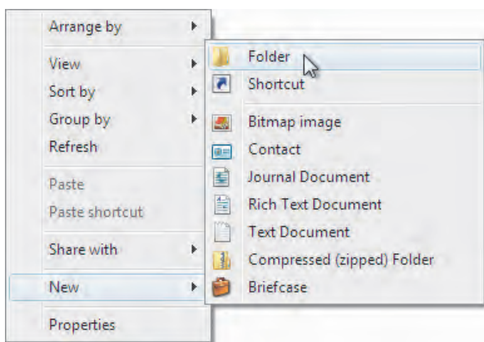


图 2.6.2　在 Windows 里，在桌面或资源管理器中右击，再选择 New → Folder

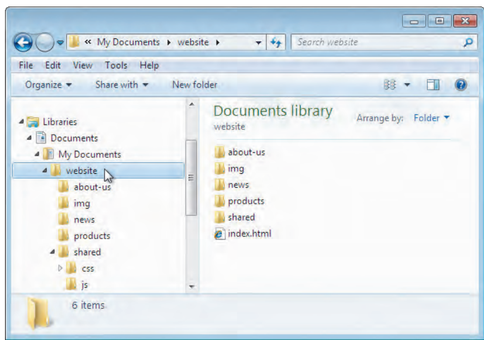


图 2.6.3　根据需要，可以将文件夹分解成多个子文件夹

提 示　使用短的、描述性的名称作为文件和文件夹的名称，使用短横线（而不用空格）分隔不同的单词。全部使用小写字母，让 URL 更容易输入，页面更容易被访问到。关于创建文件名的细节，参见 1.6 节。

2.7　在浏览器中查看网页

创建网页之后，你肯定希望看看它在浏览器中显示的样子。实际上，由于你并不知道访问者使用什么浏览器（不同的浏览器呈现页面的方式会有差异），因此建议使用多个浏览器查看页面。

在浏览器中查看网页的步骤

(1) 打开浏览器。

(2) 选择 File → Open（或 Open File、Open Page），实际叫法取决于所使用的浏览器，但不能是 Open Location（打开位置），如图 2.7.1 所示。

(3) 在弹出的对话框中，找到目标文件所在的文件夹，选中该文件，点击 Open，如图 2.7.2 所示。页面会显示在浏览器中，如图 2.7.3 所示，显示效果与发布到服务器的网页相同（关于发布网页，参见第 21 章）。对于不同的浏览器，这些步骤可能有些差异。

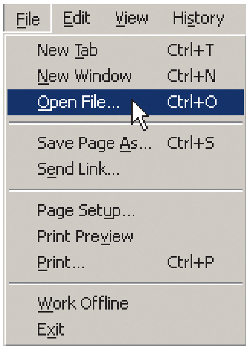


图 2.7.1　在目标浏览器（此处为 Firefox）中选择 File → Open File。Internet Explorer 中为 File → Open

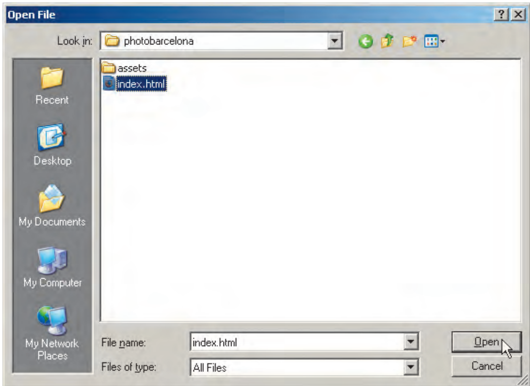


图 2.7.2　选择要打开的文件并点击 Open 按钮



图 2.7.3　页面出现在浏览器中。仔细检查一下，看看它是否符合你的预期

提 示　通常还可以双击网页图标来查看它。如果浏览器已经打开，还可以将网页图标拖到浏览器窗口来打开它。一旦掌握这些技巧，它们往往是在浏览器中查看网页的最简单方法。

提 示　一些现代浏览器没有 File→Open 这样的菜单选项来打开网页，试试上文提到的拖拽方法。

提 示　如果网页没有出现在 Open 对话框中，那么检查一下文件是否以纯文本格式保存，并且以 .html 或 .htm 为扩展名（参见 2.3 节）。

提 示　在浏览器中查看网页之前，不需要在文本编辑器中关闭文档，但一定要保存文件。在浏览器中打开网页之后，如果又在文本编辑器中对页面作了修改，那么需要再次保存文件，并在浏览器中刷新页面（可以重复在浏览器中首次打开网页的步骤，但那样做比较慢）。

提 示　在 Web 服务器上发布网站（参见第 21 章）之后，访问者才能看到网站。

2.8　借鉴他人灵感

学习其他网页开发和设计人员如何创建页面是提高 HTML 代码水平最容易的方法之一。幸好，HTML 代码很容易查看和学习。不过，文本内容、图像、音频、视频、样式表及其他外部文件可能受版权保护。通常的做法是借鉴其他人的页面为自己的 HTML 寻找灵感，再创建自己的内容。

1. 使用 View Source 查看其他设计者的 HTML 代码

(1) 在浏览器中打开网页。

(2) 选择 View Source（查看源代码，或其他浏览器中的类似选项），如图 2.8.1 和图 2.8.2 所示。这时会显示 HTML 代码，如图 2.8.3 所示。

(3) 如果愿意，可以保存该文件以作进一步研究。

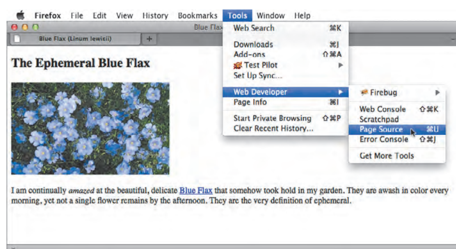


图 2.8.1　所有的桌面浏览器都有让你查看页面 HTML 源代码的菜单选项。选项的名称可能是 View Source，也可能是 Page Source（在 Firefox 中，如图），或者其他类似的名称。在 Chrome 中为 Tools→View page source（工具→查看页面源代码）

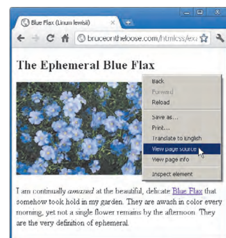


图 2.8.2　大多数浏览器还可以在页面上右击，然后在弹出的菜单中选择 View Source（或者别的类似叫法）。图上显示的是 Chrome 中的菜单。这通常是查看源代码最方便的方法，因为很难在主菜单或子菜单上找到该选项

2

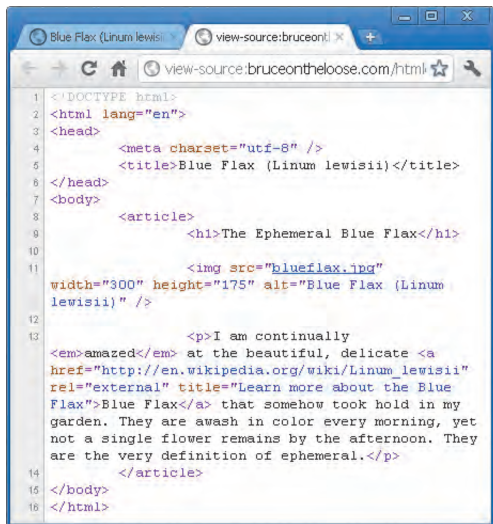


图 2.8.3　现代浏览器在其自身的选项卡或窗口中显示源代码（如图所示），而旧的浏览器可能在特定的文本编辑器中显示。页面内容的颜色与 HTML 元素、属性和属性值的颜色是区分开来的。这称为语法高亮（syntax highlighting）。左侧显示的行号并不是 HTML 代码的一部分，也并非所有的浏览器都会在查看源代码模式中显示行号。它们只是 Chrome 在查看源代码窗口中提供的一种指示

2. 通过开发者工具查看其他设计者的 HTML 代码

查看网页源代码的另一种方法是使用浏览器的开发者工具。不同浏览器提供的开发者工具并不一样，但有些功能是一样的。

这些工具提供了一种交互式的查看源代码的视图。可以审查页面特定部分的 HTML 和 CSS，在浏览器中编辑代码，并立即查看修改后的效果。可以对任何网站（不仅仅是自己的）使用开发者工具。使用开发者工具作的修改都是临时的，这些工具并未真正修改页面的 HTML 和 CSS。开发者工具是学习 HTML 和 CSS 不可多得的资料，通过它们可以查看某个特定的效果是如何实现的，可以随意改动代码而不必担心破坏任何东西。

关于开发者工具（既有现代浏览器的，也有旧浏览器的）的信息，参见 20.1 节“浏览器开发者工具”。

提 示　不存在关于谁能在万维网上发布网站的规则。这就是万维网的伟大之处——它是一种进入门槛很低的开放媒体。你可以是一名新手，一位专家，或者介于二者之间的角色。查看其他网站源代码时应该记住这一点，如果有些代码看起来并不怎么样，不要仅仅因为该网站放在网上就认为其制作者知道的比你多。有大量网站可以作为编码最佳实践的范例，也有大量网站做得并不那么理想。保持批判的态度，如果对某项技术的合理性有所怀疑，请查看本书或其他资源。

提 示　要保存代码，可以将代码从 View Source 窗口中复制到文本编辑器中，再保存文件。

提 示　要同时保存源代码及网页资源（如图像），在大多数浏览器中可以选择 File → Save As（或 File → Save Page As，文件 → 页面另存为）。不过，保存页面时，浏览器可能改写代码的某些部分，因此代码与使用上一条提示保存的代码并不完全一样。

提 示　关于查看网页的 CSS，参见 8.8 节。

第 3 章

基本 HTML 结构

3

本章内容

- 开始编写网页
- 创建页面标题
- 创建分级标题
- 理解 HTML5 的文档大纲
- 对分级标题进行分组
- 普通页面构成
- 创建页眉
- 标记导航
- 创建文章
- 定义区块
- 指定侧栏
- 创建页脚
- 创建通用容器
- 使用 ARIA 改进可访问性
- 为元素指定 class 或 id 名称
- 为元素添加 title 属性
- 添加注释

本章讨论的是构建文档基础和结构所需的 HTML 元素，即网页内容的大纲和主要的语义化容器。

你将学到以下内容：

- 开始编写网页
- HTML5 的文档大纲
- h1 ~ h6、hgroup、header、nav、article、section、aside、footer 及 div 元素（其中大多数是HTML5新增的元素）
- ARIA 的 role 属性如何提升页面的可访问性
- 为元素应用 class 和 id 属性
- 为元素应用 title 属性
- 为代码添加注释

创建清晰、一致的结构不仅可以为页面建立良好的语义化基础，也可以大大降低在文档中应用层叠样式表（CSS）的难度（从第 7 章开始讲解 CSS）。

如果你还没有这样做过，强烈建议你在继续阅读前先读读第 1 章。第 1 章展示了一个简单的 HTML 页面，并解释了一些基本概念。由于那是你第一次接触网页，因此我会在这里复述一些（但非所有的）信息，并假定你对其他信息很熟悉，可以基于它们开始构建网页。

另外，如果你读过笔者的 *The HTML Pocket Guide*，你会发现很多熟悉的材料。

3.1 开始编写网页

每个 HTML 文档都应该包含以下基本成分（如图 3.1.1 所示）：

- DOCTYPE；
- html 元素（包含 lang 属性。该属性不是必需的，但推荐加上）；
- head 元素；
- 说明字符编码的 meta 元素；

❑ title 元素；

❑ body 元素。

这份 HTML 等同于一张空白的纸，因为 body 里面没有任何内容，如图 3.1.2 所示。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title></title>
</head>
<body>


</body>
</html>
```

图 3.1.1　这是每个 HTML 页面的基础。缩进并不重要，但结构很重要。在这个例子中，默认语言（由 lang 属性设定）被设为代表英语的 en。字符编码被设为 UTF-8

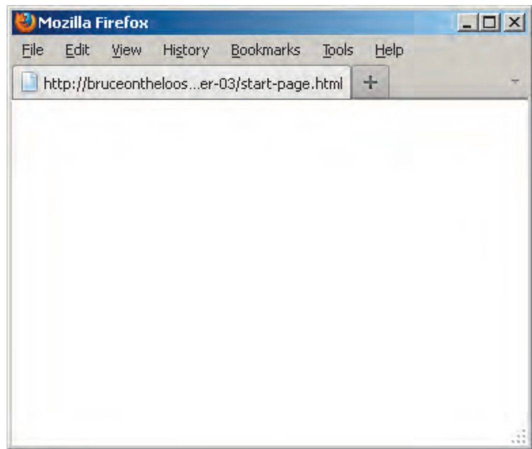


图 3.1.2　最少 HTML 基础代码在 Firefox 中显示的样子。如你所见，什么内容也没有！不过，很快就可以开始添加内容了

在添加任何内容或其他信息之前，需要先建立起页面的基础。

1. 编写 HTML5 页面开头的步骤

(1) 输入 <!DOCTYPE html>，声明页面为 HTML5 文档。（关于 HTML 早期版本的相关信息，参见本节末尾的"改进后的 HTML5 DOCTYPE"。）

(2) 输入 <html lang="*language-code*">，开始文档的实际 HTML 部分。其中，*language-code* 是页面内容默认语言的代码。例如，<html lang="en"> 表示英语，<html lang="fr"> 表示法语。可用语言代码的完整列表参见 www.bruceontheloose.com/references/language-codes.html。

(3) 输入 <head>，开始网页文档的头部。

(4) 输入 <meta charset="UTF-8"/>，将文档的字符编码声明为 UTF-8。如果愿意，也可以输入 utf-8。空格和斜杠是可选的，因此 <meta charset="UTF-8"> 也是有效的。（UTF-8 以外的其他字符编码也是有效的，不过 UTF-8 的适用面最广，很少有需要用其他编码的情况。）

(5) 输入 <title></title>。这里将包含页面的标题。3.2 节中将讲解如何添加标题文字。

(6) 输入 </head>，结束页面文档的头部。

(7) 输入 <body>，开始页面的主体。这里是放置页面内容的地方。

(8) 为页面内容预留一些空行。后续章节会讲解如何创建内容。

(9) 输入 </body>，结束主体。

(10) 输入 </html>，结束页面。

步骤似乎有点多，不过，由于所有的页面都是这样开始编写的，可以使用一个单独的 HTML 页面作为创建每个页面的模板，以节省输入的时间。实际上，很多代码编辑器都可以为新页面指定初始的代码，这样就更简单了。如果你在编辑器中没有找到 Settings（设置）或 Preferences（偏好设置）菜单，可以在 Help（帮助）中搜索。

2. 网页的两个部分：head 和 body

快速回顾一下第 1 章学到的知识，HTML

页面分为两个部分：head 和 body，参见图 3.1.1。DOCTYPE 出现在每个页面的开头，就像一本书的序言。

在文档 head 部分，通常要指明页面标题，提供为搜索引擎（如 Google）准备的关于页面本身的信息，加载样式表，以及加载 JavaScript 文件（不过，出于性能考虑，多数时候在页面底部 </body> 标记结束前加载 JavaScript 是更好的选择）。随着进一步阅读本书，你会看到一些关于这些内容的例子。除了 3.2 节就要讲解的 title，其他 head 里的内容对页面访问者来说都是不可见的。

body 元素包住页面的内容，包括文本、图像、表单、音频、视频以及其他交互式内容，也就是访问者看见的东西。本书有好几章都讲述 HTML 的内容相关元素，其中一些会在本章提前接触到。

提示 使用 HTML5 DOCTYPE 可以确保浏览器以可靠的模式呈现页面，同时告诉 HTML 验证器根据 HTML5 允许的元素和语法检查代码。第 20 章会讨论 HTML 验证器。

提示 HTML5 的 DOCTYPE 不区分大小写。例如，有的人选择输入 <!doctype html>。不过，使用 <!DOCTYPE html> 是更常规的做法，参见图 3.1.1。

提示 紧跟 DOCTYPE 的 html 元素应该包含页面的所有元素，参见图 3.1.1。

提示 要确保你的代码编辑器已配置为以 UTF-8 保存文件，与代码中通过 <meta charset="UTF-8" /> 语句指定的编码相同，参见图 3.1.1。（如果指定了另一种编码，就按那种编码保存文件。）并非所有的编辑器都默认以 UTF-8 保存文件，但大多数编辑器可以在菜单或面板中选择编码（参见 2.3 节）。如果页面没有设置为 UTF-8，有的字母（如带重音符的 i、带波形符（～）的 n）就会变成一些奇怪的字符。

提示 嵌套在 head 元素里的代码不一定要缩进，参见图 3.1.1。不过，缩进可以让你一眼看出 head 从哪里开始，包含什么内容，以及在哪里结束。在有些页面中，head 变得很长并不少见。

改进后的 HTML5 DOCTYPE

有了 HTML5 以后，编写代码的开头部分变得容易多了。HTML5 的 DOCTYPE 短得让人惊讶，特别是跟从前的 DOCTYPE 相比。

在 HTML 4 和 XHTML 1.0 时代，有好几种可供选择的 DOCTYPE，每一种都会指明 HTML 的版本，以及使用的是过渡型还是严格型模式。由于它们太难记，开发人员不得不每次都从某个地方把这些代码复制进来。

例如，下面是 XHTML 严格型文档的 DOCTYPE。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
→ xhtml1-strict.dtd">
```

看起来令人费解。

幸好，所有的浏览器，无论新旧，都理解 HTML5 的 DOCTYPE，因此你可以坚持在所有页面中使用它，忘掉过去的那些 DOCTYPE。（唯一可能用到过去那些 DOCTYPE 的情况是继承一个旧站，而网站的拥有者不允许将 DOCTYPE 修改为 HTML5 的版本。）

3.2　创建页面标题

3.1 节 HTML 基础代码中 <title></title> 仅为占位符，现在开始讨论 title 元素。

每个 HTML 页面都必须有一个 title 元素。每个页面的标题都应该是简短的、描述性的，而且是唯一的，如图 3.2.1 所示。在大多数浏览器中，页面标题出现在窗口的标题栏（Chrome 是个例外），如图 3.2.2 所示。如果支持标签浏览（所有近年发布的主流浏览器都支持），页面标题也会出现在标签上。页面标题还会出现在访问者浏览历史列表和书签里，如图 3.2.3 所示。

或许更为重要的是，页面标题会被 Google、Bing、Yahoo! 等搜索引擎采用，从而能够大致了解页面内容，并将页面标题作为搜索结果中的链接显示，如图 3.2.4 所示。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Antoni Gaudí - Introduction
    → </title>
</head>
<body>


</body>
</html>
```

图 3.2.1　title 元素必须位于 head 部分，将它放置在指定字符编码的 meta 元素后面

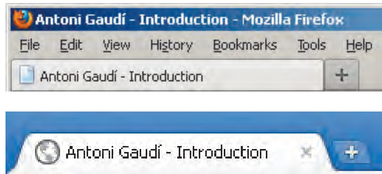


图 3.2.2　在大多数浏览器（如 Firefox）中，页面标题既显示在标题栏，也显示在标签上。不过，Chrome（下图）只在标签上显示页面标题

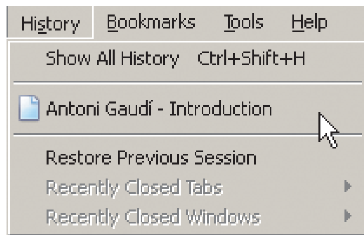


图 3.2.3　页面标题也出现在访问者的 History 面板（如图）、收藏夹列表以及书签列表中

总之，要让每个页面的 title 是唯一的，从而提升搜索引擎结果排名，并让访问者获得更好的体验。



图 3.2.4　或者最重要的是，页面标题通常是 Google 等搜索引擎的搜索结果中链接的文字，它也是判断搜索结果中页面相关度的重要因素。此处为 Google 中显示的页面标题和部分主体内容

创建页面标题的步骤

(1) 将光标放在文档 head 中的 <title> 和 </title> 之间。

(2) 输入网页的标题。

提 示　title 元素是必需的。

提 示　title 中不能包含任何格式、HTML、图像或指向其他页面的链接。

提 示　创建新页面时，有的代码编辑器会预先为页面标题填上默认文字，除非已经按照 3.1 节中介绍的方法使用了特定的开头代码。要注意到这些默认文字，确保用自己的标题替换它们。

提示 如果页面标题中包含特殊字符（如重音）或某些符号，那么它们必须是页面所用编码的一部分（如果使用 UTF-8，则一般不会产生问题），否则就必须用引用来编写它们（可用的字符实体引用列表参见 www.elizabethcastro.com/html/extras/entities.html）。同时，别忘了在保存页面时选择恰当的编码（如 UTF-8），从而确保特殊字符被正确地保存（参见 2.3 节）。

深入探讨页面标题

很多开发人员不太重视 title 元素，甚至那些相当用心、很有经验的开发人员也是这样。他们只是简单地输入网站名称，并将其复制到全站每一个网页。或者更糟糕，让 title 文字仍然保存为代码编辑器默认添加的文字。如果流量是网站追求的指标之一，这样做对建站者和潜在的访问者都会产生巨大的损害。

不同搜索引擎确定网页排名和内容索引规则的算法是不一样的。不过，title 通常都扮演着重要的角色。搜索引擎会将 title 作为判断页面主要内容的指标，并将页面内容按照与之相关的文字进行索引。有效的 title 应包含几个与页面内容密切相关的关键词。

作为一种最佳实践，选择能简要概括文档内容的文字作为 title 文字。这些文字既要对屏幕阅读器用户友好，又要有利于搜索引擎排名。其次，将网站名称放入 title（这不是必需的）。将网站名称放在 title 的开头是很常见的做法，不过将页面特有的文字放在开头是更好的做法。

建议将 title 的核心内容放在前 60 个字符（含空格）中，因为搜索引擎通常将超过此数目（作为基准）的字符截断。不同浏览器显示在标题栏里字符数的上限不尽相同，不过也不会超过 60。浏览器标签会将标题截得更短，因为它占的空间较少。

3.3 创建分级标题

HTML 提供了 6 级标题用于创建页面信息的层级关系。使用 h1 ~ h6 元素对标题进行标记，其中 h1 是最高级别的标题，h2 是 h1 的子标题，h3 是 h2 的子标题，以此类推。你将了解到，标题是页面中最重要的 HTML 元素类别之一。

为了理解 h1 ~ h6 标题，可以将它们与销售报告、学期论文、产品手册和新闻文章等非 HTML 文档里的标题进行类比。当你撰写这些文稿时，会根据需要为内容的每个主要部分指定一个标题和任意数量的子标题（以及子子标题，等等）。总之，这些标题代表了文档的大纲。网页的分级标题也是这样的，如图 3.3.1 所示。3.4 节会对此作进一步分析。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Antoni Gaudí - Introduction
    ➝ </title>
</head>
<body>

<h1>Antoni Gaudí</h1>

<h2 lang="es">La Casa Milà</h2>

<h2 lang="es">La Sagrada Família</h2>

</body>
</html>
```

图 3.3.1 用标题建立文档结构，就像大纲一样。这里，标记为 h2 的 La Casa Milà 和 La Sagrada Família 是标记为 h1 的顶级标题 Antoni Gaudí 的子标题。（lang="es" 表示这段内容为西班牙语，并不影响文档结构。）如果 La Sagrada Família 是 h3，它就成了 La Casa Milà 的子标题（也是 Antoni Gaudí 的子子标题）。标题之间的空行并不是必需的，它们不会对内容的显示产生任何影响。如果继续编写页面其余部分的代码，相关的内容（段落、图像、视频等）就要紧跟在对应的标题后面。图 3.3.2 是分级标题的例子

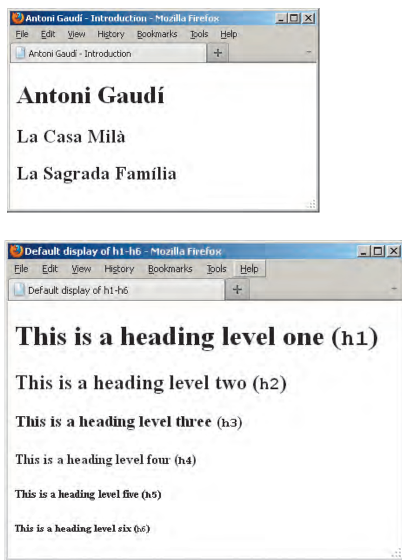


图 3.3.2　所有的标题都默认以粗体显示，h1 的字号比 h2 的大，而 h2 的又比 h3 的大，以此类推。不过，标题的外观并不是由其层级决定的，可以通过 CSS 改变它们的样式

使用标题对网页进行组织

(1) 在 HTML 的 body 部分，输入 <h*n*>，其中 *n* 是 1 ~ 6 的数字，此数字取决于要创建的标题的级别。h1 是最重要的标题，而 h6 则是最不重要的标题。

(2) 输入标题的内容。

(3) 输入 </h*n*>，其中 *n* 是与第 (1) 步中相同的数字。

> **提示**　由于 h1 ~ h6 标题对页面大纲的定义有重要的影响，因此它们非常重要。通常，浏览器会从 h1 到 h6 逐级减小标题的字号。不过别忘了要依据内容所处的层次关系选择标题级数，而不是根据你希望文字应该显示的大小。这样做能让页面具有较高的语义化程度，提升 SEO 效果和可访问性。可以按照你希望的样子为标题添加样式，包括字体、字号、颜色等。用 CSS 实现这些样式的详细说明参见第 10 章。

> **提示**　搜索引擎对标题赋予很高的权重，尤其是 h1（这并不是说页面中的 h1 越多越好，搜索引擎并不会被愚弄）。同时，屏幕阅读器用户通常通过键盘按键在标题间移动，这让他们可以快速了解页面内容并去查看最感兴趣的内容，而不用把整个页面从头到尾听完。总之，要拥有一个符合逻辑的标题层级结构。

> **提示**　在全站使用一致的标题层级以提升用户体验。

> **提示**　如果要创建指向某个标题的链接，需要在标题中添加 id（参见 6.3 节）。

> **提示**　需要说明的是，图 3.3.1 中每个 h2 中的 lang 属性表示相应的内容不是通过 **<html lang="en">** 声明的默认语言（英文），而是另一种语言（西班牙语，由代码 es 表示）。

3.4　理解 HTML5 的文档大纲

上一节讲了构成页面大纲的标题元素 h1 ~ h6。这一节将进一步讲解一些 HTML5 特有的影响文档大纲的元素。

你知道，使用标题元素可以为每个 HTML 文档定义一个基本的大纲，就像目录一样。现在，大纲还无法在页面中明确地显示出来（也许有一天浏览器会提供显示它的方法），但它提供的语义对搜索引擎和屏幕阅读器来说是非常有意义的。搜索引擎和屏幕阅读器通过大纲确定页面的结构并为用户提供相关信息。

在 HTML5 之前的 HTML 和 XHTML 中，h1 ~ h6 是创建文档大纲的全部元素。HTML5 则提供了四个分块内容（sectioning content）元素——article、aside、nav 和 section。这

些元素将文档划分为不同的区块，并定义了 h1 ~ h6（以及 header 和 footer）元素的范围。

这意味着每个分块元素都有它自己的 h1 ~ h6 层次结构，与这门语言早期的版本相比，这是一项重大的改进。同时，每个页面都可以有一个以上的 h1，而且 HTML 规范也推荐这样做（不过，下面我很快将要讲到，为什么应该限制 h1 的数量）。

上述这些都会影响大纲。让我们比较一下两种等价的大纲，看看其工作原理。在两个例子中，不妨想象每个标题之后都有众多段落和其他内容。

第一个大纲只用了标题元素，它是完全有效的 HTML5，那些拥有 HTML 和 XHTML 经验的人应该对它非常熟悉，如图 3.4.1 所示。

```
...
<body>
    <h1>Product User Guide</h1>
    <h2>Setting it Up</h2>
    <h2>Basic Features</h2>
    <h3>Video Playback</h3>
    <h2>Advanced Features</h2>
</body>
</html>
```

图 3.4.1　文档大纲版本 1

第二个版本（如图 3.4.2 所示）既用了标题元素，又用了 HTML5 的 section 元素（包含一个嵌套的 section 元素）。（注意：缩进并不重要，它不会对大纲产生影响，但它可以让元素的包含关系显得很清晰。）

早先，我提到浏览器还无法将大纲显示出来。不过，可以使用 Geoffrey Sneddon 的 HTML 5 Outliner（http://gsnedders.html5.org/outliner/）查看大纲。这是一个对文档大纲进行可视化显示的简单好用的工具。用该工具对上面的版本 1 和版本 2 生成大纲，不难发现，尽管它们的 h1 ~ h6 标题层级并不相同，但大纲是一样的：

1. Product User Guide
    1. Setting it Up
    2. Basic Features
        1. Video Playback
    3. Advanced Features

```
...
<body>
    <h1>Product User Guide</h1>
    <section>
      <h1>Setting it Up</h1>
    </section>

    <section>
      <h1>Basic Features</h1>
      <section> <!-- 嵌套的，因此是其父元
      素的子区块 -->
        <h1>Video Playback</h1>
      </section>
    </section>

    <section>
      <h1>Advanced Features</h1>
    </section>
</body>
</html>
```

图 3.4.2　文档大纲版本 2（与版本 1 的大纲相同，但是有含义的标记更多）

可以看到，版本 2 中每个 section 元素都变成离它最近的 h1 ~ h6 或分块内容祖先（这此例中，分块内容祖先仍是 section 元素）的子区块。全部四个 HTML5 分块内容元素（article、aside、nav 和 section）都会如此，即使将它们混合到一起。

相比之下，如果版本 2 没有 section 元素（可以称之为版本 3，如图 3.4.3 所示），那么大纲会很不一样。

这时每个标题的重要性都是一样的（h1），意即没有任何子标题（或子子标题）：

1. Product User Guide
2. Setting it Up

3. Basic Features
4. Video Playback
5. Advanced Features

```
...
<body>
    <h1>Product User Guide</h1>
    <h1>Setting it Up</h1>
    <h1>Basic Features</h1>
    <h1>Video Playback</h1>
    <h1>Advanced Features</h1>
</body>
</html>
```

图 3.4.3　文档大纲版本 3（与版本 1 和版本 2 的大纲并不相同）

拥有相同含义的大纲（即版本 1 和版本 2），尽管都是有效的，但版本 2 更为可取，因为 section 元素的语义更为明确。实践中，可以用一个 article 元素将版本 2 的全部内容包起来，在这种情形下，加上 article 更为合理（尽管所生成的大纲会略有不同）。图 3.4.4 展示了一个例子。

```
...
<body>
    <article>
       <h1>Product User Guide</h1>
       <section>
          <h1>Setting it Up</h1>
       </section>

       <section>
          <h1>Basic Features</h1>
          <section>
             <h1>Video Playback</h1>
          </section>
       </section>

       <section>
          <h1>Advanced Features</h1>
       </section>
    </article>
</body>
</html>
```

图 3.4.4　具有明确语义的大纲

别忘了，每个标题之后应该紧跟着相关的文本、图像和其他内容（即将在后面了解到）。不过现在这些内容都没有加进来，让你可以专注于标题和大纲。

1. 迎合当前生态系统

但是，等等！还要对代码做一项调整。还记得我说过"下面我很快将要讲到，为什么应该限制 h1 的数量"吗？尽管每个分块内容元素（article、aside、nav 和 section）都可以从 h1 开始其标题层级，但这并不是强制性的。实际上，在可预见的将来，如果分块内容元素代表的是已经存在的 h1 的子标题，则最好使用 h2 或更低级别的标题开始每个区块。

下面是这样做的原因。

在不断演进的万维网世界，有大量变化的部分。像 HTML5 这样的新规范在最终确定（需要过一些年，目前 HTML5 还没有最终确定）之前，每天都在变化。新的浏览器发布了，新版本的屏幕阅读器和其他辅助技术推出了，但它们的步调并不会完全同步。

相反，每个浏览器都会逐步添加功能（这多半是好事），但不一定是与其竞争对手相同的功能（这不太好），而且肯定不会是与其竞争对手同步的时间表。屏幕阅读器也是同样的情况。因此，尽管现代浏览器都支持大量 HTML5 特性，在本书写作时没有一个浏览器能将 HTML5 大纲呈现给屏幕阅读器，而屏幕阅读器也无法将其呈现给用户。

简而言之，这意味着屏幕阅读器和其他辅助技术还无法区分直接放在 body 里的 h1 和包含在 article、aside、nav 和 section 里面的 h1。在它们看来，这些 h1 都是顶层的 h1。Opera 布道者 Bruce Lawson 是我知道的第一个指出这一点的人（www.brucelawson.co.uk/2009/headings-in-html-5-and-accessibility/；注意这个 URL 中的其他一些信息已经过时了，因为从那

时起规范已经发生了变化。另参见他与 Remy Sharp 的新著 *Introducing HTML5*）。

然而，屏幕阅读器用户却没有时间等待万维网跟上他们的需求。它们将继续利用标题来获取页面内容概览，对页面进行导航。而有含义的标题层级关系将让这变得更为容易，使访问者拥有更好的体验。

因此，在生态系统赶上来之前，使用 h1 ~ h6 明确地表现层次关系，就像没有分块内容元素一样，会让你和你的用户感觉更好。Lawson 等人推荐这种方法，我也是。

下面看看应该怎样做，参见图 3.4.5。

```
...
<body>
<article>
    <h1>Product User Guide</h1>
    <section>
        <h2>Setting it Up</h2>
    </section>

    <section>
        <h2>Basic Features</h2>
        <section>
            <h3>Video Playback</h3>
        </section>
    </section>

    <section>
        <h2>Advanced Features</h2>
    </section>
</article>
</body>
</html>
```

图 3.4.5 版本 4（所有 4 个版本的推荐方法）

原先处于 section 元素第一层的 h1 现在是 h2。标题 Video Playback 所在的 section 嵌套在另一个 section 里面，它原先是 h1，现在是 h3。文档大纲并未改变，只是标题层级改变了。

这个例子中只有 h1 ~ h3，不过，如果需要，也可以使用 h4 ~ h6。例如，Video Playback 的子标题应该是一个 h4（视情况可加上一个分块父元素），以此类推。

记住，这种推荐做法对所有的分块内容元素（article、aside、nav 和 section）都管用，而不仅仅是例子中出现的分块内容元素。

2. 小结

如果还没有完全掌握 HTML5 的文档大纲，建议你重读一遍关于它的讨论。看起来容易，做起来难。强烈建议你创建多种测试页面，比较它们在 HTML5 Outliner 中的结果，从而更好地理解大纲算法的原理。在做实际项目时，也应使用 Outliner，确保页面结构是符合预期的。首先，确保对 HTML5 页面进行验证，消除编码错误（参见 20.5 节，http://validator.nu/ 与 http://validator.w3.org/ 两个网址均可）。

提示 不要留下这样一种印象，即必须使用一个 article，或者 section 必须（也只能）嵌套在一个 article 里面。我们讨论的例子只是使用这些元素的一种方式，而事实上，同样的内容也可以用其他的方式进行标记，它们仍然是有效的 HTML5。稍后我们会详细讲解 article 和 section，根据内容的不同，它们有一些不同的应用场景。

HTML5 大纲算法对聚合内容的好处

目前你已看到，作为分块内容元素，根据 HTML5 的大纲算法，每个 article、aside、nav 和 section 都有其自身的大纲，该大纲可以从 h1 开始，一直到 h6。

除了给文档标题增加了灵活性，这还有另外一个不那么明显的好处：当内容出现在其他页面甚至其他网站时，不会破坏父文档的大纲，其自身的大纲也完好无损。

如今，在网站之间共享内容已变得越来越普遍了。这样的例子包括新闻聚合网站、带 RSS 源的博客、Twitter 源，等等。你即将在 3.9 节中了解到，article 元素代表一个独立的容器，它可以被聚合（并非必需，只在恰当的时候可以这样做）。

想象下面的 article 显示在另一个网站的情形：

```
...
<h2>News from around the Web</h2>

<article>
  <h1>Local Teen Prefers Vinyl over Digital</h1>

  <p>A local teen has replaced all her digital tracks with vinyl. “It’s
  ➝ groovy,” she said, on the record.</p>

  <h2>Hooked after First Album</h2>
  ...
</article>
...
```

将代码放入 HTML5 Outliner，将会看到其大纲为：

1. News from around the Web
    1. Local Teen Prefers Vinyl over Digital
        1. Hooked after First Album

因此，即使 Local Teen 标题比它上面的 h2 的层级更高（h1），它仍然是 h2 的子标题，因为它包含在 h2 下面的 article 中。而 Hooked h2 则是 News h2 的子子标题，不是在平等的层级。

将 News 标题改为 h3、h4 或任何层级，大纲都会保持不变。Local Teen 和 Hooked 也是一样，只要保持 Local Teen 的标题层级比 Hooked 的高。

3.5　对分级标题进行分组

有时，一个标题有多个连续的层级，例如带有子标题、替换标题或广告语。这时将它们放进 hgroup 元素可以指明它们是相关的，如图 3.5.1 所示。每个 hgroup 都包含两个或更多的 h1 ～ h6 标题，不可放入其他元素。

在一个 hgroup 中，只有第一个最高级别的标题会出现在文档大纲中（参见 3.4 节），这也是你选择使用 hgroup 的一个决定因素。不过，需要指出，hgroup 中的所有标题都会在浏览器中显示，如图 3.5.2 所示。

对两个或更多标题进行分组的步骤

(1) 输入 <hgroup>。

(2) 输入 <hn>，其中 *n* 是 1 ～ 6 的数字，此数字取决于要创建的标题的级别。

(3) 输入标题的内容。

(4) 输入 </hn>，其中 *n* 是与第 (2) 步中相同的数字。

(5) 重复 (2) ~ (4) 步，创建所有需成为 hgroup 一部分的标题。通常，每一个后续标题的级别都应低一级（例如，从 h1 到 h2，依次类推）。

(6) 输入 </hgroup>。

```
...
<body>

<article>
    <hgroup>
        <h1>Giraffe Escapes from Zoo</h1>
        <h2>Animals Worldwide Rejoice</h2>
    </hgroup>

    <p>... [文章内容] ...</p>
</article>

</body>
</html>
```

图 3.5.1　两个相关的标题组合在一起。在此例中，h2 是文章标题 h1 的子标题。由于 Giraffe Escapes from Zoo 被标记为最高级别的标题，因此只有它出现在文档大纲里，不过这两个标题都会出现在浏览器中，如图 3.5.2 所示。类似地，如果在这个 h1 后面再加一个 h1，新的 h1 也不会出现在大纲里，就像上面的 h2 一样。由于 h2 并不出现在大纲里，文章中的下一个标题可以是 h2（而不是 h3），并且我们可以将其看做 h1“Giraffe Escapes from Zoo”的直接子标题

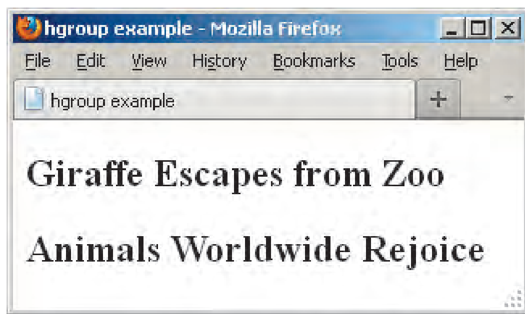


图 3.5.2　两个标题都会显示在浏览器中，就像它们没被包含在 hgroup 元素中一样

提 示　hgroup 不能仅包含一个标题，至少要包含两个。

提 示　如上所述，在一个 hgroup 中，只有第一个最高级别的标题会出现在文档大纲中，标题的顺序没有影响。因此，如果一个 hgroup 有一个 h3，后面紧跟一个 h2，那么 h2 会出现在大纲中。我们通常根据优先级对标题进行排序，因此低级别的标题（如 h3）不会出现在级别高的标题（如 h2）前面；偶尔可能遇到例外的情况。

3.6　普通页面构成

你肯定已经访问过了大量像图 3.6.1 中显示的那种网站。抛开内容不谈，我们可以看到该页面有四个主要组件：带导航的页头、显示在主体内容区域的文章、显示次要信息的侧栏以及页脚，如图 3.6.2 所示。

现在，在没有引入 CSS 的情况下，你还无法像图 3.6.1 中那样为页面添加样式，无法对页面进行排布，如图 3.6.1 和图 3.6.2 所示。你将从第 7 章开始学习 CSS，从第 10 章开始学习如何对文字进行格式化，如何添加颜色等，在第 11 章学习多栏布局。

不过，应用到这些常规页面结构的语义都是非常相似的，与布局无关。本章剩余章节的大部分内容都会讲解这些结构。按照从页面顶端向下的顺序，将依次讲解用 header、nav、article、section、aside 和 footer 定义页面的结构，以及用以添加额外样式信息或实现其他目的的通用容器 div。除了 div 以外，这些元素都是 HTML5 推出后才有的。在前面的代码示例和讨论中，你已经见到了其中的一些元素。

在学习这些元素的过程中，不要关心它们在示例布局中的位置，而应该关注它们的语义。

接下来，我们还会提前看到其他一些元素，如 ul（无序列表）和 a（链接）。这些将在后续章节讲到。



图 3.6.1　一个普通的布局，顶部是主导航，左侧是主要内容，右侧是侧栏，底部是页脚。要让页面成为这个样子，需要添加 CSS

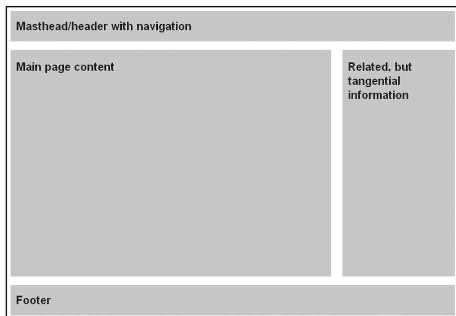


图 3.6.2　页面的常规信息类型。这只是一种安排方式，不过是很常见的一种

3.7　创建页眉

如果页面中有一块包含一组介绍性或导航性内容的区域，应该用 header 元素对其进行标记。

一个页面可以有任意数量的 header 元素，它们的含义可以根据其上下文而有所不同。例如，处于页面顶端或接近这个位置的 header 可能代表整个页面的页眉（有时称做页头），如图 3.7.1 所示。通常，页眉包括网站标志、主导航（如图 3.7.2 所示）和其他全站链接，甚至搜索框。这无疑是 header 元素最常见的使用形式，不过不要误认为是唯一的形式。

```
...
<body>
<header>
    <nav>
      <ul>
        <li><a href="#gaudi">Barcelona's
        → Architect</a></li>
        <li lang="es"><a href="#sagrada-
        → familia">La Sagrada Família</a>
        → </li>
        <li><a href="#park-guell">Park
        → Guell</a></li>
      </ul>
    </nav>
</header>
</body>
</html>
```

图 3.7.1　这个 header 代表整个页面的页眉。它包含一组代表整个页面主导航的链接（在 nav 元素中）。出于提高可访问性的目的，可以将可选的 role="banner" 应用到整页页眉。图 3.7.3 中显示了这样做的例子



图 3.7.2　包含导航的页眉

header 也很适合对页面深处的一组介绍性或导航性内容进行标记。例如，一个区块的目录，如图 3.7.3 所示。

```
...
<body>
<header role="banner">
    ... [网站标识、导航等] ...
</header>

<article>
    <header>
       <h1>Frequently Asked Questions</h1>
       <nav>
          <ul>
             <li><a href="#answer1">What is your return policy?</a><li>
             <li><a href="#answer2">How do I find a location?</a><li>
             ...
          </ul>
       </nav>
    </header>

    <!-- header的链接指向这里 -->
    <article id="answer1">
       <h2>What is your return policy?</h2>
       <p> ... [答案] ... </p>
    </article>

    <article id="answer2">
       <h2>How do I find a location?</h2>
       <p> ... [答案] ... </p>
    </article>
    ...
</article> <!--结束父元素article -->

</body>
</html>
```

图 3.7.3 这个页面有两个 header，一个是整个页面的，另一个是 Frequently Asked Questions 父元素 article 的。注意第一个并不包含任何 h1 ~ h6 标题，而第二个则有。关于第一个 header 中出现的可选的 role 属性，参见本节最后一条提示

header 元素是 3.4 节中提到的四个分块内容元素中的一个。这意味着在考虑文档大纲时，位于 header 内的任何 h1 ~ h6 标题都会被认为处于 header 的范围中，而不是整个页面。因此，header 通常包含其自身的标题（h1 ~ h6 或 hgroup），但这并不是强制性的。例如，图 3.7.3 中有标题，但图 3.7.1 中就没有。

创建页眉的步骤

(1) 将光标放置在需要创建页眉的元素里。

(2) 输入 <header>。

(3) 输入页眉内容，包括各种类型的内容，它们分别由各自的 HTML 元素（大多数将在本书余下部分讲到）进行标记。例如，header 可以包含 h1 ~ h6 标题、标志、导航、搜索框，等等。

(4) 输入 </header>。

提 示　只在必要时使用 header。大多数情况下，如果只有 h1 ～ h6 或 hgroup，没有其他需要与之组合在一起的伴生内容，就没有必要用 header 将它包起来。

提 示　header 与 h1～h6 元素中的标题（参见 3.3 节）是不能互换的。它们都有各自的语义目的。

提 示　不能在 header 里嵌套 footer 元素或另一个 header，也不能在 footer 或 address 元素里嵌套 header。

提 示　header 不一定要像示例那样包含一个 nav 元素（如图 3.7.1 和图 3.7.3 所示），不过在大多数情况下，如果 header 包含导航性链接，就可以用 nav。在图 3.7.3 中，nav 包住 Frequently Asked Questions 链接列表是恰当的，因为它是页面内的主要导航组（将在 3.8 节中讨论）。

提 示　要了解 header 如何取代 HTML5 之前版本中 div 元素的一个功能，参见 3.13 节。

提 示　要了解如何在 header 中使用 role="banner"，参见 3.14 节。

3.8　标记导航

HTML 的早期版本没有元素明确表示主导航链接的区域，而 HTML5 则有这样一个元素，即 nav。nav 中的链接可以指向页面中的内容，如图 3.8.1 所示，也可以指向其他页面或资源，或者两者兼而有之。无论是哪种情况，应该仅对文档中重要的链接群使用 nav。

```
...
<body>
<header>
    <nav role="navigation">
        <ul>
            <li><a href="#gaudi">Barcelona's
            ➝ Architect</a></li>
            <li lang="es"><a href="#sagrada-
            ➝ familia">La Sagrada Família</a>
            ➝ </li>
            <li><a href="#park-guell">Park
            ➝ Guell</a></li>
        </ul>
    </nav>
</header>
</body>
</html>
```

图 3.8.1　这些链接（a 元素）代表一组重要的导航，因此将它们放入一个 nav 元素。通常用一个 ul 元素（无序列表）对链接列表进行标记，除非链接是面包屑链接。如果是面包屑，则使用一个 ol 元素（有序列表）。更多有关列表的信息参见第 15 章。role 属性并不是必需的，不过它可以提高可访问性。更多有关向 nav 应用 role="navigation" 的信息参见本节最后一个提示

如果你仔细看过上一节的代码，就已经见识过 nav 元素了。下面将那一段代码搬过来，并对 nav 进行了突出显示（参见图 3.8.1）。nav 元素不会对其内容添加任何默认样式，如图 3.8.2 所示。



图 3.8.2　在默认情况下，我们的导航看起来相当普通。那些圆点并不是由 nav 元素产生的。除了开启一个新行以外，该元素没有任何默认样式。显示这些圆点是因为每个链接都包在一个 li 元素（列表项）里面。使用 CSS 可以隐藏这些圆点或显示其他的东西，还可以将这些链接水平显示，改变它们的颜色，让它们看起来像按钮，等等。本书将从第 7 章开始讲解 CSS

将一组链接指定为重要导航

(1) 输入 <nav>。

(2) 输入一组链接并将其标记为 ul（无序列表）结构，除非链接的顺序比较重要（例如面包屑链接）。如果链接顺序重要，就将其标记为 ol（有序列表）结构。（要了解链接和列表，分别参见第 6 章和第 15 章。）

(3) 输入 </nav>。

提示 有编写 HTML 或 XHTML 经历的开发人员或许已经习惯了使用 ul 或 ol 元素对链接进行结构化。在 HTML5 中，nav 并没有取代这种最佳实践。应该继续使用这些元素，只是在它们的外围简单地包上一个 nav（参见图 3.8.1）。

提示 尽管屏幕阅读器整体上还在追赶 HTML5 新出现的语义功能，但 nav 能帮助它们识别页面的主导航，并允许用户通过键盘直接跳至这些链接。这可以提高页面的可访问性，提升访问者的体验。

提示 HTML5 规范不推荐对辅助性的页脚链接（如“使用条款”、“隐私政策”等）使用 nav，这不难理解。不过，有时页脚会再次显示顶级全局导航，或者包含“商店位置”、“招聘信息”等重要链接。在大多数情况下，我们推荐将页脚中的此类链接放入 nav 中。

提示 HTML5 不允许将 nav 嵌套在 address 元素中。

提示 要了解如何在 nav 中使用 role="navigation"（参见图 3.8.1），请参见 3.14 节。

3

深入了解 nav

上文提到，要在页面中插入一组链接并不意味着一定要将它们包含在 nav 中。

下面的示例新闻页面包含四个链接列表，其中只有两个列表具有足够的重要性，可以包在 nav 中。（可以看到，一些代码片段被缩减了。）

```
...
<body>
   <header>
      <!-- 站点标识可以放在这里 -->
      <!-- 全站导航 -->

      <nav>
         <ul> ... </ul>
      </nav>
   </header>

   <div id="main">
      <h1>Arts &amp; Entertainment</h1>
```

```
        <article>
            <h1>Gallery Opening Features the Inspired, Inspiring</h1>
            <p>... [故事内容] ... </p>

            <aside>
                <h1>Other Stories</h1>

                <!-- 不包在nav中-->
                <ul> ... [故事链接] ... </ul>
            </aside>
        </article>
    </div>
    <aside id="sidebar">
        <nav><!-- 次级导航 -->
            <ul>
                <li><a href="/arts/movies/">Movies</a></li>
                <li><a href="/arts/music/">Music</a></li>
                ...
            </ul>
        </nav>
    </aside>

    <footer>
        <!-- 辅助性链接并未包在nav中 -->
        <ul> ... </ul>
    </footer>
</body>
</html>
```

位于 aside（参见 3.11 节）中的次级导航允许用户跳转到 Arts & Entertainment 目录中的其他页面，因此它构成了页面的一个主要导航区块。不过，Other Stories 这个 aside 则不是。

那么，如何判断是否对一组链接使用 nav 呢？归根结底，这取决于内容的组织情况。至少，应该将网站全局导航（让用户可以跳至网站各个主要部分的导航）标记为 nav。这种 nav 通常（但并不总是）出现在页面级的 header 元素里面（参见 3.7 节）。

3.9　创建文章

HTML5 的另一个新元素便是 article，参见图 3.9.1。你已经见过一些用到该元素的示例。现在，让我们深入了解一下它的作用。

根据其名称，你大概会猜想 article 用于包含像报纸文章一样的内容。不过，article 并不局限于此。在 HTML5 中，“文章”（article）更接近于“项目”（item）。

HTML5 对该元素的定义如下。

article 元素表示文档、页面、应用或网站中一个独立的容器，原则上是可独立分配或可再用的，即聚合。它可以是一篇论坛帖子，一篇杂志或报纸文章，一篇博客文章，一则用户提交的评论，一个交互式的小部件或小工具，或者任何其他独立的内容项。

其他 article 的例子包括电影或音乐评论、案例研究、产品描述，等等。你或许惊讶于它还可以是交互式的小部件或小工具，不过这些确实是独立的、可再分配的内容项。

```
...
<body>
<header>
    <nav role="navigation">
      ... [包含链接的ul] ...
    </nav>
</header>
<article>
    <h1 id="gaudi">Barcelona's Architect</h1>

    <p>Antoni Gaudí's incredible buildings
    ➝ bring millions of tourists to
    ➝ Barcelona each year.</p>

    <p>Gaudí's non-conformity, already
    ➝ visible in his teenage years, coupled
    ➝ with his quiet but firm devotion to
    ➝ the church, made a unique foundation
    ➝ for his thoughts and ideas. His
    ➝ search for simplicity, based on his
    ➝ careful observations of nature, are
    ➝ quite apparent in his work, from the
    ➝ <a href="#park-guell">Park Guell</a>
    ➝ and its incredible sculptures and
    ➝ mosaics, to the Church of the <a href=
    ➝ "#sagrada-familia">Sacred Family</a>
    ➝ and its organic, bulbous towers.</p>

    <h2 id="sagrada-familia" lang="es">La
    ➝ Sagrada Família</h2>

    ... [图像和段落] ...

    <h2 id="park-guell">Park Guell</h2>

    ... [图像和段落] ...
</article>

</body>
</html>
```

图 3.9.1 为了精简，我对文章内容进行了缩写，略去了与上一节相同的 nav 代码。可以在本书网站查看完整的代码（www.bruceontheloose.com/htmlcss/examples/）。在这个例子里只有段落和图像，不过 article 可以包含各种类型的内容，如视频、图形、列表等。这段代码生成的页面如图 3.9.2 所示



图 3.9.2 现在，页面有了 header、nav 和 article 元素，以及它们各自的内容。在不同的浏览器中，article 中标题的字号可能不同。可以通过 CSS 使它们在不同的浏览器中显示相同的大小（参见第 10 章）

创建文章的步骤

(1) 输入 <article>。

(2) 输入文章的内容。内容可以包含任意数量的元素。元素类型包括段落、列表、音频、视频、图像、图形等。

(3) 输入 </article>。

提 示 正如你在 3.4 节中学到的，article 是四个分块内容元素中的一个，另外三个是 nav、section 和 aside。

提 示 可以将 article 嵌套在另一个 article 里面，只要里面的 article 与外面的 article 是部分与整体的关系。不过，不能将 article 嵌套在 address 元素里面。

提示　一个页面可以有多个 article 元素（也可以没有）。例如，博客的主页通常包括几篇最新的文章，其中每一篇都是其自身的 article。

提示　一个 article 里包含一个或多个 section 元素并不是强制性的。让 article 里的 h1 ～ h6 独自存在也是很好的，不过定义 section 则会让文章的语义更明显。每个 section 都可以有自己的标题层级关系（参见 3.4 节）。

提示　article 和 section 元素很容易相互混淆，这也是我直接引用 HTML5 有关定义的原因。我不希望你将它们的意思理解偏。我会在 3.10 节中讨论 section 以及如何在这两个元素之间作出选择。

提示　要了解在哪种情况下可对 article 使用 role="main"，参见 3.14 节。在图 3.9.1 中的 article 放入这段代码是合理的，因为该元素是页面主要内容的容器，不过我有意略去了，以避免给读者留下所有 article 元素都需要 role="main" 的印象。

更多 article 示例

图 3.9.1 中的示例只是使用 article 的一种方式，下面看看其他的用法。

例 1（基本文章）

```
<article>
   <h1>The Diversity of Papua New Guinea</h1>
   <p>Papua New Guinea is home to more than 800 tribes and languages ...</p>
   ... [故事内容的其余部分] ...

   <footer> <!-- article的页脚，而非页面的页脚 -->
      <p>Leandra Allen is a freelance journalist who earned her degree in
      ➝ anthropology from the University of Copenhagen.</p>
      <address>
      You may reach her at <a href="mailto:leandra@therunningwriter.com">
      ➝ leandra@therunningwriter.com</a>.
      </address>
   </footer>
</article>
```

注意 footer 和 address 元素的使用（对它们的讨论分别参见本章和第 4 章）。这里，address 只应用于其父元素 article，而非整个页面或任何包含在那个 article 里面的 article（如例 2 中的读者评论）。

例 2 展示了嵌套在 article 父元素里面的 article 元素。该例中嵌套的 article 是用户提交的评论，就像你在博客或新闻站上见到的评论部分。该例还显示了 section 元素（参见 3.10 节）和 time 元素（在第 4 章讨论）的用法。

例 2（嵌套文章）

```
<article>
    <h1>The Diversity of Papua New Guinea</h1>
    ... [父元素文章内容] ...

    <footer>
       ... [父元素文章页脚] ...
    </footer>

    <section>
       <h2>Reader Comments</h2>
       <article>
          <footer>travelgal wrote on <time datetime="2011-11-17"
          ➝ pubdate>November 17, 2011</time>:</footer>
          <p>Great article! I've always been curious about Papua New
          ➝ Guinea.</p>
       </article>
       <article>
          ... [下一篇读者评论] ...
       </article>
    </section>
</article>
```

这些只是使用 article 及有关元素的几个常见方式。

3.10 定义区块

article 元素有一个语义性弱一些的“近亲”元素——section，它也是 HTML5 的新元素。

HTML5 对该元素的定义如下。

section 元素代表文档或应用的一个一般的区块。在这里，section 是具有相似主题的一组内容，通常包含一个标题，如图 3.10.1 所示。

section 的例子包含章节、标签式对话框中的各种标签页、论文中带编号的区块。比如网站的主页可以分成介绍、新闻条目、联系信息等区块。

article 和 section 元素极为相似。如果对如何区分这两个元素仍存疑惑，参见“如何在 article 和 section 中作出选择”。

定义区块的步骤

(1) 输入 <section>。

(2) 输入区块的内容。内容可以包含任意数量的元素。元素类型包括段落、列表、音频、视频、图像、图形等。

(3) 输入 </section>。

提示 正如你在 3.4 节中学到的，section 是四个分块内容元素中的一个，另外三个是 nav、article 和 aside。

提示 默认情况下，你在查看页面时无法看到应用 section 后的变化（article 也是这样），不过重要的是在文档中使用这些元素可以增强语义性，参见图 3.10.2。当然，你可以使用 CSS 对 section 和 article 元素添加样式。

```
...
<body>
<header>
    <nav role="navigation">
       ... [包含链接的ul] ...
    </nav>
</header>

<article>
    <h1 id="gaudi">Barcelona's Architect</h1>

    <p>Antoni Gaudí's incredible buildings
    → bring millions of tourists to
    → Barcelona each year.</p>

    ... [另一个介绍性段落] ...

    <section>
       <h2 id="sagrada-familia" lang="es">La
       → Sagrada Família</h2>

       <p><img src="img/towers.jpg"
       → width="75" height="100" alt=
       → "Sagrada Família Towers" /> The
       → complicatedly named and curiously
       → unfinished masterpiece that is
       → the Expiatory Temple of the
       → Sacred Family is the most visited
       → building in Barcelona. In it, Gaudí
       → combines his vision of nature and
       → architecture with his devotion
       → to his faith. The Sagrada Família
       → attracts even the non-religious to
       → its doors in large part due to its
       → tragic story and its still
       → unfinished state, of which the
       → everpresent scaffolding and cranes
       → are permanent reminders.</p>
    </section>

    <section>
       <h2 id="park-guell">Park Guell</h2>

       ... [另一个图像和段落] ...
    </section>
</article>
</body>
</html>
```

图 3.10.1　这段代码与前面的代码相比，只是对 article 中介绍文字后面两个部分分别用了一个 section 包起来，其他部分完全相同。为了简洁，我再次对代码进行了缩写

提 示　记住，section 不是一个像 div 一样的通用容器，因为 section 表达一定的含义，而 div 则没有任何语义上的含义（参见 3.13 节）。

提 示　本章有好几个例子可以帮助你理解如何（以不同的方式）使用 article 和 section。

提 示　要了解在哪种情况下可对 section 使用 role="main"，参见 3.14 节。



图 3.10.2　现在，页面有了 header、nav、article 和 section 元素，以及它们各自的内容。添加 section 元素之后，默认的显示效果不会发生改变

如何在 article 和 section 中作出选择

我特意引用了 HTML5 对 section 和 article 的定义（参见 3.9 节）帮助你理解它们的区别，因为它们的区别有时很细微。不妨这样考虑：你的内容是适合用做聚合的一块独立的内容或一个小组件吗？如果是，就使用 article（否则，在大多数情况下，使用 section；另参见 3.13 节了解什么时候用 div 代替）。这并不意味着你必须聚合或分发 article 内容，只是其内容适合这样做。

如果你仍然觉得 article 和 section 有时看起来很相似，另担心，不只你自己有这种感觉，即便是经验丰富的开发人员有时也会将这两个元素用错。

正如我在第 1 章提到过的，对内容进行标记时，并非总能分出对和错，只是大多数时候有正确的选择。而其他时候，就只能依靠个人对最适合描述内容的 HTML 元素的感觉了。

在 article 和 section 之间作选择时，一定要仔细考虑，不过也不必每次都对是否用对感到担心。有时，些许主观并不会影响页面正常工作。而且，也不会有人半夜来敲你的门。

哦，我可能会，不过这只是因为外面很黑，很吓人。

3

没有 article 的 section 示例

目前你已见过几个嵌套在 article 里面的 section 的例子，参见图 3.10.1。但那只是该元素的一种用法。

下面的例子来自 HTML5 的规范（略有改动）。在这个例子中，你会看到没有 article 的 section 的应用。（你还会提前看到有序列表的应用。第 15 章会讲到 ol 和其他的列表元素。）

```
...
<body>
  <h1>graduation program</h1>

  <section>
    <h2>Ceremony</h2>
    <ol>
      <li>Opening Procession</li>
      <li>Speech by Valedictorian</li>
      <li>Speech by Class President</li>
      <li>Presentation of Diplomas</li>
      <li>Closing Speech by Headmaster</li>
    </ol>
  </section>

  <section>
    <h2>Graduates (alphabetical)</h2>
    <ol>
      <li>Molly Carpenter</li>
      <li>Anastasia Luccio</li>
      <li>Ebenezar McCoy</li>
      <li>Karrin Murphy</li>
      <li>Thomas Raith</li>
      <li>Susan Rodriguez</li>
    </ol>
  </section>
</body>
</html>
```

3.11　指定侧栏

有时，你有一部分内容与页面的主体内容并不那么相关，但可以独立存在，如图 3.11.1 所示。如何在语义上表示出来呢？

```
...
<body>

<header>
    <nav role="navigation">
       ... [包含链接的ul] ...
    </nav>
</header>

<article>
    <h1 id="gaudi">Barcelona's Architect</h1>
    ... [介绍性段落] ...

    <section>
       <h2 id="sagrada-familia" lang="es">La Sagrada Família</h2>
        ... [图像和段落] ...
    </section>

    <section>
       <h2 id="park-guell">Park Guell</h2>
       ... [另一个图像和段落] ...
    </section>
</article>

<aside role="complementary">
    <h1>Architectural Wonders of Barcelona</h1>

    <p>Barcelona is home to many architectural wonders in addition to Gaudí's work. Some of them
    ➝ include:</p>
    <ul>
       <li lang="es">Arc de Triomf</li>
       <li>The cathedral <span lang="es">(La Seu)</span></li>
       <li lang="es">Gran Teatre del Liceu</li>
       <li lang="es">Pavilion Mies van der Rohe</li>
       <li lang="es">Santa Maria del Mar</li>
    </ul>

    <p>Credit: <a href="http://www.barcelona.de/en/barcelona-architecture-buildings.html"
    ➝ rel="external"><cite>Barcelona.de</cite></a>.</p>
</aside>

</body>
</html>
```

图 3.11.1　这个 aside 是有关巴塞罗那建筑奇迹的信息，与页面主要关注的 Antoni Gaudí 内容并不那么相关，但仍可以独立存在。我可以将它嵌套在 article 里面，因为它们是相关的，但我还是决定把它放在 article 后面，让其看起来像侧栏，aside 里面的 role="complementary" 是可选的，但它可以提高可访问性。参见最后一条提示

在 HTML5 之前，一直无法显式地做到这一点。在 HTML5 中，我们有了 aside 元素，如图 3.11.2 所示。

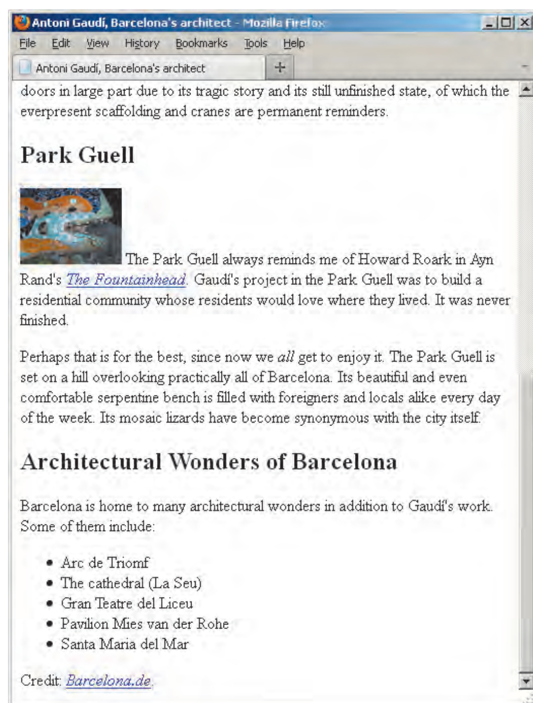


图 3.11.2　aside 出现在文章的下面，因为在 HTML 中 aside 就跟在 article 的后面，参见图 3.11.1。可以看到，浏览器在默认情况下并未对 aside 应用任何特殊样式（除了在新行开始）。不过，可以通过 CSS 控制其外观显示，参见图 3.11.3

很容易将 aside 看做侧栏，如图 3.11.3 所示，但 aside 元素其实可以用在页面的很多地方。在哪儿使用它依上下文而定。它可以是主要内容中的一个框，或者与主要内容位于同一栏而并未嵌套，或者位于（或作为）侧栏。aside 的例子包括抬升式引用、侧栏、新闻站上列出相关文章的链接框、广告、nav 元素组（如博客的友情链接）、Twitter 源、商业站上相关产品列表。



图 3.11.3　对已完成的页面应用 CSS 时，可让 aside（以 Architectural Wonders of Barcelona 开头）显示在主要内容的旁边，而不是下边。这个例子已将 aside 做成了侧栏（我们将在第 11 章学习如何制作两栏布局）

指定侧栏的步骤

(1) 输入 <aside>。

(2) 输入侧栏的内容。内容可以包含任意数量的元素。元素类型包括段落、列表、音频、视频、图像、图形等。

(3) 输入 </aside>。

提 示　尽管 aside 的一种用法是标记侧栏的内容，参见图 3.11.3，但 aside 元素本身并不影响页面的布局，参见图 3.11.2。

提 示　如果在侧栏中使用一个或多个 aside（或将其作为侧栏使用），应在 HTML 中将它们放在页面主要内容的后面，参见图 3.11.1。将重要的内容放在前面有利于 SEO 和提升可访问性。可以通过 CSS 改变它们在浏览器中显示的顺序。

提示　对于与内容有关的图像（如图表、图形或带有说明文字的插图），使用 figure（参见第 4 章）而非 aside。

提示　HTML5 不允许将 aside 嵌套在 address 元素内。

提示　要了解如何在 aside 中使用 role="complementary"，参见 3.14 节。

其他 aside 示例

上面提到，aside 可以与主要内容位于同一栏，可以嵌套在主要内容里，也可以嵌套在侧栏里。

例 1 展示了嵌套在相关内容中的 aside。

例 1（嵌套在主要内容中）

```
...
<body>
<article>
  <h1>The Diversity of Papua New Guinea</h1>
  ... [文章内容] ...
  <aside>
    <h1>Papua New Guinea Quick Facts</h1>
    <ul>
      <li>The country has 38 of the 43 known birds of paradise</li>
      <li>Though quite tropical in some regions, others occasionally
      → experience snowfall.</li>
      ...
    </ul>
  </aside>
  ... [更多文章内容] ...
</article>
</body>
</html>
```

这个 article 也可以包含一个抬升式引用，这也应该包在 aside 里。也可以弄一个“相关故事”aside，包含一组指向关于该国或周边地区（印度尼西亚、澳大利亚等）其他文章的链接。另外，aside 还可以位于页面的另外一栏，而不是嵌套在 article 里。

我们已看到了 aside 位于侧栏的例子（图 3.11.1 和图 3.11.3）。现在，考虑一套设计作品或一组案例研究的例子，其中每个 HTML 页面展示一个项目，并在毗邻的一栏里（显示位置由 CSS 控制，而不是由例 2 中的代码控制）提供指向其他项目页面的链接（嵌套在 nav 里）。

例 2（aside 并未嵌套在主要内容里，且包含一个 nav）

```
...
<body>
```

```
<!-- 页面的主体内容 -->
<article>
  <h1>... [项目名称] ...</h1>
  <figure>... [项目图片] ...</figure>
  <p>... [项目信息] ...</p>
</article>

<!-- 这个aside并未嵌套在article中 -->
<aside>
  <h1>Other Projects</h1>
  <nav>
    <ul>
      <li><a href="habitat-for-humanity.html">Habitat for Humanity
      ➝ brochure</a></li>
      <li><a href="royal-philharmonic.html">Royal Philharmonic Orchestra
      ➝ site</a></li>
      ...
    </ul>
  </nav>
</aside>
</body>
</html>
```

将这个 aside 嵌套在项目 article 里也是极好的选择，因为它们是相关的。

3

3.12 创建页脚

当你想到页脚的时候，你大概想的是页面底部的页脚。HTML5 的 footer 元素的功能不止这一个，而是像 header 一样，可以用在其他地方。

footer 元素代表嵌套它的最近的 article、aside、blockquote、body、details、fieldset、figure、nav、section 或 td 元素的页脚。只有当它最近的祖先是 body 时，它才是整个页面的页脚，参见图 3.12.1 和图 3.12.2。如果一个 footer 包着它所在区块（如一个 article）的所有内容，它代表的是像附录、索引、版权页、许可协议这样的内容。

创建页脚的步骤

(1) 放光标放在希望创建页脚的元素里。

(2) 输入 <footer>。

(3) 输入页脚的内容。

(4) 输入 </footer>。

提 示 页脚通常包含关于它所在区块的信息，如指向相关文档的链接、版权信息、作者及其他类似条目。参见“其他 footer 示例”中的例 1 和例 2。

提 示 页脚并不一定要位于所在元素的末尾，不过通常是这样的。

提 示 不允许在 footer 里嵌套 header 或另一个 footer。同时，也不能将 footer 嵌套在 header 或 address 元素里。

```
...
<body>
<header>
    <nav role="navigation">
       ... [包含链接的ul] ...
    </nav>
</header>

<article>
    <h1 id="gaudi">Barcelona's Architect</h1>
    ... [介绍性段落] ...

    <section>
       <h2 id="sagrada-familia" lang="es">La
       ➝ Sagrada Família</h2>
        ... [图像和段落] ...
    </section>

    <section>
       <h2 id="park-guell">Park Guell</h2>
       ... [另一图像和段落] ...
    </section>
</article>

<aside role="complementary">
    <h1>Architectural Wonders of Barcelona
    ➝ </h1>
    ... [侧栏的其余内容] ...
</aside>

<footer>
    <p><small>&copy; Copyright 2011</small>
    ➝ </p>
</footer>

</body>
</html>
```

图 3.12.1　这个 footer 代表整个页面的页脚，因为它最近的祖先是 body 元素。现在，页面有了 header、nav、article、section、aside 和 footer 元素。并非每个页面都需要全部这些元素，但它们确实代表了 HTML 中可用的主要页面构成要素

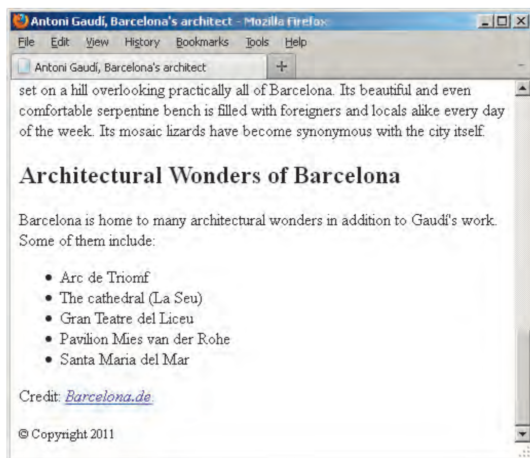


图 3.12.2　footer 元素本身不会为文本添加任何默认样式。这里，版权信息的字号比普通文本的小，这是因为它嵌套在用于对法律文本进行语义化的 small 元素里（参见第 4 章）。像其他内容一样，可以通过 CSS 修改字号

提 示　要了解 footer 如何取代 HTML5 之前版本中 div 元素的一个功能，参见 3.13 节。

提 示　要了解在哪种情况下可对 footer 使用 role="contentinfo"，参见 3.14 节。在图 3.12.1 中的 footer 放入这段代码是合理的，因为它代表的是整个页面的页脚，不过我有意略去了，避免给读者留下所有 footer 元素都需要 role="contentinfo" 的印象。参见“其他 footer 示例”中区分与正确使用 role 的示例。

其他 footer 示例

我们已经学习了整个页面 footer 的一个例子，参见图 3.12.1 和图 3.12.2。下面是另一个页面的 footer，其中的内容更多一些。

例 1（整个页面的页脚）

```
...
<body>
... [页眉和内容] ...

<!-- 这是页面的页脚，因为body是其最近的祖先 -->
<footer role="contentinfo">
  <p><small>&copy; Copyright 2011 The Corporation, Inc.</small></p>

  <ul>
    <li><a href="terms-of-use.html">Terms of Use</a></li>
    <li><a href="privacy-policy.html">Privacy Policy</a></li>
  </ul>
</footer>
</body>
</html>
```

下面的例子展示了位于一个页面区块（这里是 article）里的一个 footer，以及作为整个页面页脚的另一个 footer。（对 address 元素的解释，参见“更多 article 示例”。）

例 2（作为页面区块和整个页面的页脚）

```
...
<body>
...
<article>
  <h1>... [文章标题] ...</h1>
  <p>... [文章内容] ...</p>
  <!-- 文章的页脚 -->
  <footer>
    <p>Leandra Allen is a freelance journalist who earned her degree
    ➝ in anthropology from the University of Copenhagen.</p>
    <address>
    You may reach her at <a href="mailto: leandra@therunningwriter.
    ➝ com">leandra@therunningwriter.com</a>.
    </address>
  </footer>
</article>

<!-- 页面的页脚 -->
<footer id="footer-page" role="contentinfo">
  ... [版权信息、使用条款、隐私政策等] ...
</footer>
</body>
</html>
```

页脚中的 id="footer-page"（可以指定任何有效的 id）是可选的，其目的是将其与另一个 footer 区分开，从而能为二者分别设置样式。注意只有整个页面的页脚有 role="contentinfo"。参见 3.14 节了解此功能。

3.13　创建通用容器

有时需要在一段内容外围包一个容器，从而可以为其应用 CSS 样式或 JavaScript 效果。如果没有这个容器，页面就会不一样，参见图 3.13.1。不过，当评估内容，考虑使用 article、section、aside、nav 等元素的时候，却发现它们从语义上讲都不合适。你真正需要的是一个通用容器，一个完全没有任何语义含义的容器。这个容器就是 div（来自 division 一词）元素，参见图 3.13.2。有了 div，就可以为其添加样式（如图 3.13.3 所示）或 JavaScript 效果了。一定要阅读补充材料，了解什么时候可以在页面中使用 div。



图 3.13.1　在不使用 div 元素的情况下，我们实现了这个设计。不过，为页面内容加上 div 以后（如图 3.13.2 所示），我们有了一个可以添加更多样式的通用容器（参见图 3.13.3 显示的效果）

```
...
<body>

<div>
    <header>
        <nav role="navigation">
            ... [包含链接的ul] ...
        </nav>
    </header>

    <article>
        <h1 id="gaudi">Barcelona's Architect
        → </h1>
        ... [介绍性段落] ...

        <section>
            ... [标题、图像和段落] ...
        </section>

        <section>
            ... [标题、另一图像和段落] ...
        </section>
    </article>

    <aside role="complementary">
        <h1>Architectural Wonders of
        → Barcelona</h1>
        ... [侧栏其余内容] ...
    </aside>

    <footer>
        ... [版权信息] ...
    </footer>
</div>

</body>
</html>
```

图 3.13.2　现在有一个 div 包着所有的内容。页面的语义没有发生改变，但现在我们有了一个可以用 CSS 添加样式的通用容器，参见图 3.13.3



图 3.13.3　div 元素自身没有任何默认样式，除了从新的一行开始出现，参见图 3.13.4。不过，你可以对 div 添加样式以实现你的设计。这里，我对 div 添加了浅蓝色的背景和盒阴影。这样，我可以将 body 元素的背景改为紫色，使内容凸显出来。我还为 aside 添加了细边框。你可以看到我如何实现这个页面的 HTML 和 CSS（www.bruceontheloose.com/htmlcss/examples/）

创建通用容器的步骤

(1) 输入 <div>。

(2) 创建容器的内容，内容可以包含任意数量的元素。

(3) 在容器的结尾处输入 </div>。

提 示　像 header、footer、article、section、aside、nav、h1 ~ h6、p 和其他很多元素一样，div 在默认情况下自动显示在新的一行。

提 示　div 对使用 JavaScript 实现一些特定的交互行为或效果也是有帮助的。例如，在页面中展示一张照片或一个对话框，同时让背景页面覆盖一个半透明的层（这个层通常是一个 div）。

提 示　尽管我始终强调 HTML 用于对内容的含义进行描述，但 div 并不是唯一没有语义价值的元素。span 是与 div 对应的一个元素：div 是块级内容的无语义容器，而 span（写做 这里是内容 ）则是短语的无语义容器，例如在段落元素 p 之内。参见第 4 章了解 span。

提 示　要了解如何在 div 中使用地标角色，参见 3.14 节。

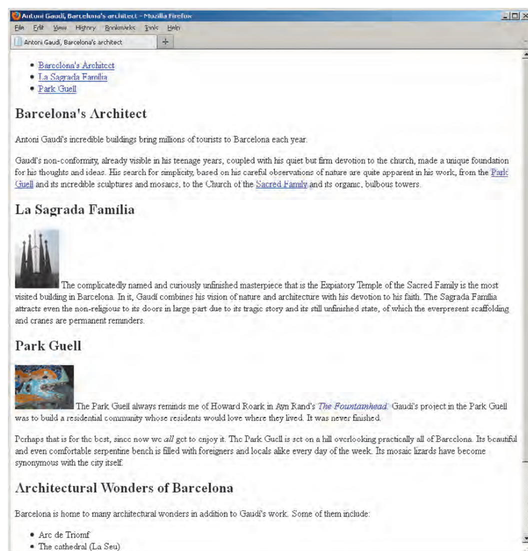


图 3.13.4　同一个页面在未对 div、标题、段落等所有元素添加 CSS 的情况下的样子。可以看到，div 并未产生任何独特的样式

有关 div 的一些历史以及何时在 HTML5 中使用它

在本章讲到的结构性元素中，div 是除 h1 ~ h6 以外唯一早于 HTML5 出现的元素。在 HTML5 之前，div 是包围大块内容（如页眉、页脚、主要内容、插图、侧栏等）从而可用 CSS 为之添加样式的不二选择。之前 div 没有任何语义含义，现在也一样。

这就是 HTML5 引入 header、footer、article、section、aside 和 nav 的原因。这些类型的构造块在网页中普遍存在，因此它们可以成为具有独立含义的元素。在 HTML5 中，div 并没有消失，只是用得上它的场合变得少了。

让我们看几个使用 div 的常见示例。

你已经见过一个了：为了添加样式，用一个容器将整个页面包起来（参见图 3.13.2 和图 3.13.3）。

如何用 div 实现两栏布局呢？我对 article 和 aside 元素分别添加了一些 CSS，让它们各自成为一栏（我们将从第 7 章开始学习 CSS；关于用 CSS 进行布局，参见第 11 章）。

然而，大多数情况下，每一栏都有不止一个区块的内容。例如，主要内容区第一个 article 下面可能还有另一个 article（或 section、aside，等等）。又如，也许你想在第二栏再放一个 aside 显示指向关于 Gaudí 的其他网站的链接，或许再加一个其他类型的元素。

你需要将希望出现在同一栏的内容包在一个 div 里，参见图 3.13.5，然后对这个 div 添加相应的样式。（你或许在想是否也可以用 section，但要知道该元素并不是用做添加样式的通用容器。）图 3.13.6 有助于你形象地理解代码和希望呈现的 CSS 布局之间的关系。记住这只是这段 HTML 可能形成的布局之一。要知道 CSS 是相当强大的。

因此，将希望在样式上组成一栏的内容用 div 包起来是一种非常常见的做法（当然，也可以用 div 包住两个以上的栏）。至于放在其中的内容，则可能差别极大，视页面内容安排而定。别忘了，作为内容区块的主要语义化容器，article、section、aside 和 nav 可以位于任何地方。此外，header 和 footer 也是这样。在本例中，主内容区只有 article，侧栏只有 aside，但不要对此作过分解读。

不过，可以肯定的是，div 应该作为你最后想到的容器，因为它没有任何语义价值。大多数时候，使用 header、footer、article、section、aside 甚至 nav 代替 div 会更合适。但是，如果语义上不合适，也不必为了刻意避免使用 div 而使用上述元素。div 有用得上的地方，只是需要限制其使用。

说到这里，有一种情况，div 适合于所有（或几乎所有，这由你决定）页面容器，而不是使用新的 HTML5 元素。更多信息参见 11.3 节。

```
...
<body>

<!-- 开始页面容器 -->
<div id="container">
    <header>
       ...
    </header>

    <!-- 应用CSS后的第一栏 -->
    <div id="content">
       <article>
          ...
       </article>

       <article>
          ...
       </article>

       ... [所需的其他区块] ...
    </div>
    <!-- 结束第一栏 -->

    <!-- 应用CSS后的第二栏 -->
    <div id="sidebar">
       <aside>
          ...
       </aside>

       <aside>
          ...
       </aside>

       ... [所需的其他区块] ...
    </div>
    <!-- 结束第二栏 -->

    <footer>
       ...
    </footer>
</div>
<!-- 结束页面容器 -->

</body>
</html>
```

图 3.13.5　这个页面有一个包含整个页面的 div，另外还新增了两个。一个带有 id="content" 的 div 集合主体内容，使之可以添加样式成为第一栏。另一个带有 id="sidebar" 的 div 包着希望显示为第二栏的内容。然后，可以在 CSS 中使用 id 找到对应的 div 并为其添加样式

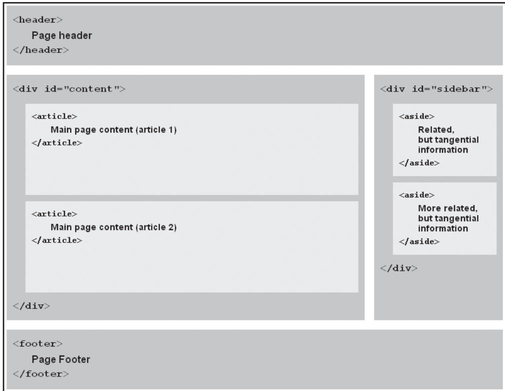


图 3.13.6　此图显示了图 3.13.5 中代码如何在概念上对应到 CSS 布局对应的形式。这是很常见的一种安排，但只是针对这段 HTML 众多可能的 CSS 样式中的一种。一定要阅读 3.14 节，了解如何进一步增强页面的语义和可访问性

3.14　使用 ARIA 提升可访问性

在 1.2 节的“为什么语义很重要”小节中我们已经了解到，对内容用最适合描述它的元素进行标记可以提升可访问性。如果你一直在这样做，非常好。在这一节，我会讲到如何在 HTML 中添加一些简单的属性进一步提升网站的可访问性。

WAI-ARIA（Web Accessibility Initiative’s Accessible Rich Internet Applications，无障碍网页倡议之可访问的富互联网应用，也简称 ARIA）是一种技术规范，自称“有桥梁作用的技术”。之所以这样说，是因为在 HTML 提供相应的语义功能之前，它会使用属性来填补一些语义上的空白。

例如，如果你想让屏幕阅读器用户知道如何跳至（或跳过）页面的主要内容，应该使用什么 HTML 标记呢？如果是要跳至搜索框，又该使用什么标记呢？你即将了解到，虽然 ARIA 和 HTML5 之间有一些重合（它们都致力于填补一些语义上的空白），

但 HTML5 尚没有提供上述两个问题的解决方法。然而，ARIA 的地标角色（landmark role）可以做到。地标角色可以帮助用户识别页面区域，包括 application、banner、complementary、contentinfo、form、main、navigation 和 search。

尽管地标角色和 HTML5 元素之间有一些重合，但目前屏幕阅读器对 ARIA 的支持更多。因此你可以继续按既有的方式创建 HTML，同时添加一些 ARIA 角色提升页面的可访问性。在图 3.14.1 中，我对 3.13 节中的示例代码添加了 ARIA 地标角色和一个 nav 元素。尽管我对每个 aside 元素都添加了 complementary 角色，但是使用 <div id="sidebar" **role=**"complementary"> 对整个侧栏标记 complementary 角色也同样是有效的。在对整个 div 标记 complementary 角色之前，要确保其中所有的内容都符合 complementary 内容的标准。

下面是 ARIA 规范中对几种地标角色的定义，以及推荐的用法。图 3.14.1 中的代码和图 3.14.2 的代码实现演示了使用这些地标角色的效果，与 3.13 节中的类似。

- role="banner"（横幅）

  包含面向全站内容而非页面特有内容的区域。

  面向全站的内容通常包括网站标志、网站赞助者标志、全站搜索工具等。横幅通常显示在页面的顶端，而且通常横跨整个页面的宽度。

  用法：将其添加到页面级的版头（通常为 header 元素），每个页面只用一次。

- role="navigation"（导航）

  指向文档内不同部分或相关文档的导航性元素（通常为链接）的集合。

  用法：与 HTML5 的 nav 元素相似，应将其添加到每个 nav 元素，或其他包含导航性链接的容器。这个 role 可在每个页面上使用多次。

```
...
<body>

<!-- 开始页面容器 -->
<div id="container">
    <header role="banner">
      ...
      <nav role="navigation">
        ... [包含链接的ul] ...
      </nav>
    </header>

    <!-- 应用CSS后的第一栏 -->
    <div id="content" role="main">
      <article>
        ...
      </article>

      <article>
        ...
      </article>

      ... [所需的其他区块] ...
    </div>
    <!-- 结束第一栏 -->

    <!-- 应用CSS后的第二栏 -->
    <div id="sidebar">
      <aside role="complementary">
        ...
      </aside>

      <aside role="complementary">
        ...
      </aside>

      ... [所需的其他区块] ...
    </div>
    <!-- 结束第二栏 -->

    <footer role="contentinfo">
      ...
    </footer>
</div>
<!-- 结束页面容器 -->

</body>
</html>
```

图 3.14.1　3.13 节中的示例代码，新增了一个 nav 元素和 5 个不同的地标角色

- ❑ role="main"（主体）

  页面的主体内容。

  用法：将其添加到内容主体部分的容器。这个容器通常是一个 div 元素，也可能是一个 article 或 section。除在极少数情况下，一个页面应该只有一个标记为 main 的区域。

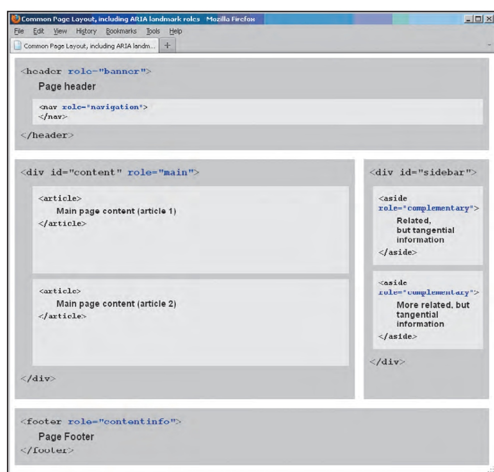


图 3.14.2　这是 3.13 节中的布局示意图，不过现在添加了 ARIA 角色。正文中提到，也可以将 role="complementary" 加到侧栏 div，而非每一个 aside 元素

- ❑ role="complementary"（补充性内容）

  文档中作为主体内容补充的支撑部分。它对区分主体内容是有意义的。complementary 角色表明其包含的内容与主体内容是相关的。

  用法：与 HTML5 的 aside 元素相似，应将其添加到每个 aside 或包含补充性内容的div。这个role可在每个页面上使用多次。

- ❑ role="contentinfo"（内容信息）

  包含关于文档的信息的大块可感知区域。这类信息包括版权声明和指向隐私权声明的链接等。

  用法：将其添加至整个页面的页脚（通常为 footer 元素）。

总之，在 HTML 中添加 ARIA 地标角色通常是个好主意。我在本书其他示例和配套网站中也使用了它们。再说一遍，如果没有它们，页面也照常工作，但加入它们可以改善一些用户的体验。提示中列出的屏幕阅读器测试结果可以帮你决定是否应该使用它们（结果支持除 Window-Eyes 7.5 以外的屏幕阅读器）。

提 示　对表单元素来说，form 角色是多余的。search 用于对搜索框进行标记（BBC、Yahoo! 和 Google 有时用到该角色，同时也用到了其他的一些地标角色）。application 为高级用法。

提 示　地标角色只是 ARIA 规范（www.w3.org/TR/wai-aria/）众多特性的一种。你或许对这份实施指南感兴趣：www.w3.org/WAI/PF/aria-practices/。

提 示　无障碍访问倡导者 Steve Faulkner 和 Jason Kiss 分别发表了屏幕阅读器对地标角色支持情况的测试结果（分别位于 www.html5accessibility.com/tests/landmarks.html 和 www.accessibleculture.org/research/html5-aria-2011/）。Faulkner 发表的相关讨论参见 www.paciellogroup.com/blog/2011/11/latest-aria-landmark-support-data/ 和 www.paciellogroup.com/blog/2011/07/html5-accessi-bility-chops-aria-landmark-support/。

提 示　NVDA（Windows 软件，可从 www.nvda-project.org 免费下载）、VoiceOver（Mac OS X 和 iOS 4+ 自带软件）和 JAWS（Windows，可在 www.freedomscientific.com 下载试用版）是最先进的几款屏幕阅读器。强烈建议你至少试用其中的一种，从而更好地理解语义化 HTML 如何影响屏幕阅读器用户的体验。最好将屏幕阅读器测试列入你的日常开发流程。

提示　可以在 CSS 选择器中使用 ARIA 角色属性。实际上，通过使用恰当的地标角色，可以将 id="content" 和 id="sidebar" 属性从代码示例中移除（如图 3.14.1 所示），参见第 11 章。

3.15　为元素指定类或 ID 名称

尽管并非必需，但是可以给 HTML 元素分配唯一的标识符，或指定其属于某个（或某几个）类别，也可以同时指定标识符和类别。然后，就可以对具有给定 id 或 class 名称的元素添加样式了。这是它们最常见的用法，但并非唯一的用法（参见本节提示部分）。

1. 为元素添加唯一标识符的方法

在元素的开始标记中输入 id="name"，其中 name 是唯一标识该元素的名称，参见图 3.15.1。name 几乎可以是任何字符，只要不以数字开头且不包含空格。

2. 为元素指定类别的方法

在元素的开始标记中输入 class="name"，其中 name 是类别的名称，参见图 3.15.1。如果要指定多个类别，用空格将不同的类别名称分开即可，如 class="name anothername"（可以指定两个以上的类别名称）。

提示　HTML 文档中的每个 id 都必须是唯一的。换句话说，一个页面里不能出现两个具有相同 id 的元素，并且每个元素都只能有一个 id。相同的 id 可以出现在不同的页面里，同一 id 也不一定每次都赋予同一元素，尽管这是惯常的做法。

提示　相反，一个 class 名称可以分配给页面中任意数量的元素，并且一个元素可以有一个以上的 class。

class 属性和微格式

认为 class 属性只能为一组元素应用 CSS 样式是一种常见的误解。事实情况并非如此，该属性也可以在不引入额外元素的情况下增强 HTML 的语义。

微格式就可以做到这一点。微格式使用约定的 class 名称标识一块 HTML，如事件或基于时间的活动（hCalendar 微格式），或标识人、组织和公司（hCard），或描述人际关系（XFN[①]）。这些只是当前定义的众多微格式中的少数几种，而更多的微格式还在生产过程中。

应用程序、搜索机器人和其他软件可以理解和使用 HTML 中的微格式。例如，Firefox 插件 Operator 可以将任何给定页面的微格式呈现出来。

关于微格式的使用，参见 http://microformats.org。

① XFN 是 XHTML Friends Network（XHTML 社交网络）的缩写。——译者注

```
...
<body>

<div id="container">
    <header>
        <nav role="navigation">
            <ul id="toc">
                <li><a href="#gaudi">Barcelona's Architect</a></li>
                <li><a href="#sagrada-familia" lang="es">La Sagrada Família</a></li>
                <li><a href="#park-guell">Park Guell</a></li>
            </ul>
        </nav>
    </header>

    <article class="architect" role="main">
        <h1 id="gaudi">Barcelona's Architect</h1>

        <p>Antoni Gaudí's incredible buildings bring millions of tourists to Barcelona each year.</p>
        ...

        <section class="project">
            <h2 id="sagrada-familia" lang="es">La Sagrada Família</h2>
            ...
        </section>

        <section class="project">
            <h2 id="park-guell">Park Guell</h2>
            ...
        </section>
    </article>
    ...
</div>
</body>
</html>
```

图 3.15.1 为元素添加唯一的 id 属性，就可以在对其进行格式化、添加指向该元素的链接或附加 JavaScript 行为的时候标识该元素。为一个或多个元素添加 class 属性，就可以对所有这类元素统一进行格式化。例如，architect 和 project 类别可以应用到关于其他建筑师的内容，从而为它们应用一致的格式。nav 中的链接指向 h1 和 h2 中的 id（关于链接，参见第 6 章）。其他的 id 是为格式化而创建的。关于 id 及相关的示例参见 3.13 节。id 和 class 属性不会影响元素的外观，除非在 CSS 中引用它们

提示 class 和 id 属性可以添加到任何元素。一个元素可以既有 id 又有多个 class。

提示 关于通过 id 或 class 为元素添加样式，参见 9.3 节。

提示　不管你打算如何使用 id 和 class，都应该为它们选择有意义的（即语义化的）名称。例如，在使用出于格式化目的的 class 时，应避免使用描述表现样式的名称，如 class="red"，因为你可能在下周决定将配色方案改为蓝色。尽管在 CSS 中对分配给某一类元素的颜色进行修改是相当容易的，但这样做会导致你的 HTML 拥有一个名为红色却实际呈现另一种颜色的 class。同时，改变 HTML 中所有的 class 通常是一项繁琐的工作。

提示　考虑为元素选择 class 还是 id 时，如果是出于样式化目的，则通常使用 class，因为这样就可以使用同一 class 将相同的样式应用到其他元素。不过，也有希望直接对某个元素（以及任何可能的后代）添加样式的时候，这时就可以用 id。

提示　id 属性会自动将该元素转化成锚，从而可以创建指到这里的链接。详细解释参见 6.3 节。

提示　可以使用 class 属性实现微格式（参见“class 属性和微格式”）。

提示　可以使用 JavaScript 访问 id 和 class 属性，从而为特定的元素添加行为。

3.16　为元素添加 title 属性

可以使用 title 属性（不要与 title 元素混淆）为网站上任何部分加上提示标签，参见图 3.16.1 和图 3.16.2。不过，它们并不只是提示标签。屏幕阅读器可以为用户朗读 title 文本，因此使用 title 可以提升可访问性。

```
...
<body>
    <header role="banner">
       <nav role="navigation">
          <ul id="toc" title="Table of
          Contents">
             <li><a href="#gaudi" title=
             ➝ "Learn about Antoni
             ➝ Gaudí">Barcelona's
             ➝ Architect</a></li>
             <li><a href="#sagrada-familia"
             ➝ lang="es">La Sagrada Família
             ➝ </a></li>
             <li><a href="#park-guell">Park
             ➝ Guell</a></li>
          </ul>
       </nav>
    </header>

    ...
</body>
</html>
```

图 3.16.1　可以为任何元素添加 title，不过用的最多的是链接



图 3.16.2　当访问者将鼠标指向加了标签的元素时，就会显示 title。如果指向 Barcelona's Architect 链接，会看到 Learn about Antoni Gaudí，因为该链接有它自身的 title 属性

在网页中为元素添加标签

在要添加 title 标签的 HTML 元素中，输入 title="*label*"，其中 *label* 是访问者将鼠标移到这个元素上时希望出现在提示框里的文本，或者希望由屏幕阅读器朗读的文本。

提示　旧版本的 Internet Explorer（IE7 及之前的版本）还会将 img 元素（参见第 5 章）的 alt 属性作为提示框的文本。不过，如果 img 元素同时出现 title 和 alt 属性，则提示框会采用 title 属性的内容，而不是 alt 属性的内容。

3.17　添加注释

可以在 HTML 文档中添加注释，标明区块开始和结束的位置，提醒自己（或未来的编辑者）某段代码的意图，或者阻止内容显示出来等，参见图 3.17.1。这些注释只会在用文本编辑器或浏览器的“查看源代码”选项打开文档时显示出来。访问者在浏览器中是看不到它们的，如图 3.17.2 所示。


```
...
<body>
    ...

    <!-- ==== 开始文章 ==== -->
    <article class="architect">
       <h1 id="gaudi">Barcelona's Architect</h1>

    <!-- 这一段不会显示，因为它被注释掉了
       <p>Antoni Gaudí's incredible buildings bring millions of tourists to Barcelona each
       ➝ year.</p>
    -->

       <p>Gaudí's non-conformity, already visible in his teenage years, coupled with his quiet
       ➝ but firm devotion to the church, made a unique foundation for his thoughts and ideas. His
       ➝ search for simplicity ...</p>

       <section class="project">
          <h2 id="sagrada-familia" lang="es">La Sagrada Família</h2>
          ...
       </section>

       <section class="project">
          <h2 id="park-guell">Park Guell</h2>
          ...
       </section>
    </article>
    <!-- 结束文章 -->

    <!--
     待处理事项：在页面上线之前添加另一篇关于其他著名建筑的文章

    -->

    ...
</body>
</html>
```

图 3.17.1　这段示例代码包括四个注释。其中有两个一起标记了文章的开始和结束位置。另一个“注释掉”了第一段，使其不在页面中显示出来（如果希望永久性地移除该段，最好在 HTML 中将它删掉）。最后一个注释用以提醒制作者在将网站上线之前再添加一些内容。要确保网站上线前将这些临时注释（如“待处理事项”）移除，防止访问者查看源代码时看到它们。更多示例参见 3.13 节



图 3.17.2　注释是不可见的（尽管在查看源代码时会看见它们）。类似地，如果将一些内容“注释掉”，它们也不会显示出来（参见图 3.17.1）。代码里包含的第一个段落在这里并没有显示

在 HTML 页面中添加注释的步骤

(1) 在 HTML 文档中希望插入注释的位置，输入 <!--。

(2) 输入注释。

(3) 输入 --> 结束注释文本。

提示　注释很适合用以提醒自己（或未来的编辑者）引入、移除或更新某些区块，参见图 3.17.1。

提示　注释的另一个用法是记录版本号。

提示　在主要区块的开头和结尾处添加注释是一种常见的做法，这样可以让你或一起合作的开发人员将来修改代码变得更加容易。笔者喜欢使用一种比结束注释更为醒目的格式编写开始注释，从而更容易区分这两个位置。

提示　在发布网站之前，应该用浏览器查看一下加了注释的页面。这样能帮你避免由于弄错注释格式导致私人化的注释内容直接暴露给公众访问者的情况。

提示　对特别私人化的注释要格外小心。尽管通常通过浏览器访问页面不会看见注释内容，但通过浏览器的“查看源代码”功能就能看见它们，如果用户将网页保存为 HTML 源代码，他们也能看见这些注释。

提示　注释不能嵌套在其他注释里。

提示　这里演示的注释语法仅适用于 HTML。CSS 和 JavaScript 的注释语法与此并不相同。CSS 和 JavaScript 均使用 /* 注释内容 */ 对一行或多行进行注释，此外 JavaScript 还可以用 // 注释内容进行单行注释。

第 4 章 文本

本章内容

- ❑ 开始新的段落
- ❑ 添加作者联系信息
- ❑ 创建图
- ❑ 指定时间
- ❑ 标记重要或强调的文本
- ❑ 指明引用或参考
- ❑ 引述文本
- ❑ 突出显示文本
- ❑ 解释缩写词
- ❑ 定义术语
- ❑ 创建上标和下标
- ❑ 标注编辑和不准确的文本
- ❑ 标记代码
- ❑ 使用预格式化的文本
- ❑ 指定细则
- ❑ 创建换行
- ❑ 创建 span
- ❑ 其他元素

除非网站充满视频和图片，否则网页的大部分内容还是文本。本章将说明针对不同的文本类型，尤其是（但不仅仅是）那些句子或短语里的文本，应该选择哪些合适的 HTML 语义化元素。

例如，em 元素用于标识强调的文本，cite 元素用于标识对艺术作品、电影、图书等内容的引用。

浏览器通常会为这些文本元素添加不同的样式，以区别于普通文本。例如，em 和 cite 元素中的文本都是斜体的。又如，code 元素专门用于对脚本或程序中的代码进行格式化，该元素中的文本默认使用等宽字体。

内容显示的样子与为其使用的标记没有关系。因此，不应该为了让文字变为斜体就使用 em 或 cite。添加样式是 CSS 的事情。

相反，应该选择能描述内容的 HTML 元素。如果浏览器默认添加的样式与你想用 CSS 设置的样式相同，那只不过是一种额外的奖励。如果不相同，直接自己编写 CSS 覆盖默认样式就可以了。

4.1 开始新的段落

HTML 会忽略你在文本编辑器中输入的回车和其他额外的空格。要在网页中开始一个新的段落，应该使用 p 元素，参见图 4.1.1 和图 4.1.2。

```
…
<body>

<article>
    <h1>Antoni Gaudí</h1>
    <p>Many tourists are drawn to
    ➝ Barcelona to see Antoni Gaudí's
    ➝ incredible architecture.</p>

    <p>Barcelona celebrated the 150th
    ➝ anniversary of Gaudí's birth in
    ➝ 2002.</p>

    <h2>La Casa Milà</h2>
    <p>Gaudí's work was essentially useful.
    ➝ <span lang="es">La Casa Milà</span> is
    ➝ an apartment building and real people
    ➝ live there.</p>

    <h2>La Sagrada Família</h2>
    <p>The complicatedly named and curiously
    ➝ unfinished Expiatory Temple of the
    ➝ Sacred Family is the most visited
    ➝ building in Barcelona.</p>
</article>

</body>
</html>
```

图 4.1.1　毫不奇怪，p 是最常用到的 HTML 元素之一



图 4.1.2　这是段落通常默认的呈现方式。使用 CSS 可以控制所有的内容元素的格式

开始新段落的步骤

(1) 输入 <p>。

(2) 输入新段落的内容。

(3) 输入 </p> 结束这个段落。

提 示　可以为段落添加样式，包括字体、字号、颜色等。详细内容参见第 10 章。

提 示　要控制行间距，参见 10.6 节。要控制段后间距，参见 11.8 节或 11.9 节。

提 示　可以通过 CSS 改变段落文本的对齐方式，包括左对齐、右对齐和居中对齐（参见 10.13 节）。

4.2　添加作者联系信息

你或许以为 address 元素是用于标记邮政地址的，但其实并非如此（有一种例外的情况，参见提示）。实际上，没有专门用于标记邮件地址的 HTML 元素。

相反，address 元素是用以定义与 HTML 页面或页面一部分（如一篇报告或新文章）有关的作者、相关人士或组织的联系信息，通常位于页面底部或相关部分内，参见图 4.2.1 和图 4.2.2。

提供作者联系信息

(1) 如果要为一个 article 提供作者联系信息，就将光标放在该元素内。如果要提供整个页面的作者联系信息，就将光标放在 body 中（更常见的做法是放在页面级的 footer 里）。

(2) 输入 <address>。

(3) 输入作者的电子邮件地址、指向联系信息页的链接等。

(4) 输入 </address>。

```
...
<body>

<article>
    <h1>Museum Opens on the Waterfront</h1>
    <p>The new art museum not only introduces
    → a range of contemporary works to the
    → city, it's part of larger development
    → effort on the waterfront.</p>
    ... [故事内容的其余部分] ...

    <!-- 文章的页脚，其中包含文章的地址信息
    → -->
    <footer>
        <p>Tracey Wong has written for <cite>
        → The Paper of Papers</cite> since
        → receiving her MFA in art history
        three years ago.</p>
        <address>
        Email her at <a href="mailto:
        → traceyw@thepaperofpapers.com">
        → traceyw@thepaperofpapers.com
        → </a>.
        </address>
    </footer>
</article>

<!--页面的页脚，其中包含整个页面的地址信息
→ -->
<footer>
    <p><small>&copy; 2011 The Paper of
    → Papers, Inc.</small></p>

    <address>
    Have a question or comment about the
    → site? <a href="site-feedback.html">
    → Contact our Web team</a>.
    </address>
</footer>

</body>
</html>
```

图 4.2.1　这个页面有两个 address 元素：一个用于 article 的作者，另一个位于页面级的 footer 里，用于整个页面的维护者。注意 article 的 address 只包含联系信息。尽管 article 的 footer 里也有关于 Tracey Wong 的背景信息，但这些信息是位于 address 元素外面的

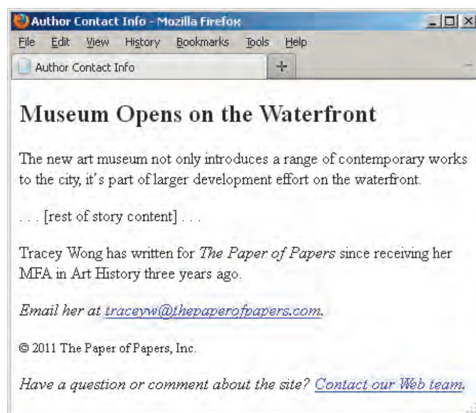


图 4.2.2　address 元素中的文字默认以斜体显示

提 示　大多数时候，联系信息的形式是作者的电子邮件地址或指向联系信息页的链接。联系信息也有可能是作者的邮政地址，这时将地址用 address 标记就是有效的。但是用 address 标记公司网站“联系我们”页面中的办公地点，则是错误的用法。

提 示　如果 address 嵌套在 article 里，则属于其所在的最近的 article 元素的祖先；否则属于页面的 body。说明整个页面的作者的联系信息时，通常将 address 放在 footer 元素里。

提 示　article 里的 address 提供的是该 article 作者的联系信息（参见图 4.2.1），而不是嵌套在该 article 里的其他任何 article（如用户评论）的作者的联系信息。

提 示　address 只能包含作者的联系信息，不能包括其他内容，如文档或文章的最后修改时间（参见图 4.2.1）。此外，HTML5 禁止在 address 里包含以下元素：h1 ~ h6、article、address、aside、footer、header、hgroup、nav 和 section。

提示　关于 article 的 footer 元素，参见第 3 章。

4.3　创建图

在出版行业，图和文字通常是相伴出现的。图可以是图表、图形、照片、插图、代码片段，等等。你很容易在报纸、杂志、报告等地方看到它们。本书的多数页面里也都有图。

在 HTML5 出现之前，没有专门实现这个目的的元素，因此一些开发人员想出了他们自己的解决办法（通常用到了不那么理想的、没有语义的 div 元素）。通过引入 figure 和 figcaption，HTML5 改变了这种情况。根据定义，figure 指的是由文档主要内容引用的一块独立的内容（具有可选的标题）参见图 4.3.1 和图 4.3.2。可选的 figcaption 是 figure 的标题，出现在 figure 内容的开头或结尾处（参见 4.3.1）。

创建图及其标题的步骤

(1) 输入 <figure>。

(2) 作为可选步骤，输入 <figcaption> 开始图的标题。

(3) 输入标题文字。

(4) 如果在第 (2)、(3) 步创建了标题，就输入 </figcaption>。

(5) 通过添加图像、视频、数据表格等内容的代码以创建图。

(6) 如果没有在 figure 内容之前包含 figcaption，则可以在内容之后重复第 (2) ~ (4) 步。

(7) 输入 </figure>。

提示　通常，figure 是引用它的内容的一部分，参见图 4.3.1，但它也可以位于页面的其他部分，或位于其他页面（如附录）。

```
...
<body>

<article>
    <h1>2011 Revenue by Industry</h1>
    ... [报告内容] ...

    <figure>
      <figcaption>Figure 3: 2011 Revenue
      ➝ by Industry</figcaption>

      <img src="chart-revenue.png"
      ➝ width="180" height="143" alt=
      ➝ "Revenue chart: Clothing 42%,
      ➝ Toys 36%, Food 22%" />
    </figure>

    <p>As Figure 3 illustrates, ... </p>

    ... [更多报告内容] ...
</article>

</body>
</html>
```

图 4.3.1　这个 figure 只有一个图表，不过放置多个图像或其他类型的内容（如数据表格、视频等）也是允许的。figcaption 元素并不是必需的，但如果包含它，它就必须是 figure 元素里的第一个或最后一个元素。figure 没有默认样式，除了在现代浏览器中会出现在新的一行

提示　figure 元素可以包含多个内容块。例如，图 4.3.1 中可以包含两个图表：一个表示收入，一个表示利润。不过要记住，不管 figure 里有多少内容，只允许有一个 figcaption。

提示　不要简单地将 figure 用做在文本中嵌入独立内容实例的方法。这种情况下，通常更适合用 aside 元素（参见 3.11 节）。

提示　在 HTML5 中，figure 元素也是一种区块根，这意味着它可以有 h1 ~ h6 标题（及其自身的大纲），但文档大纲不会包含这些标题。这与分块内容是不一样的，参见 3.4 节。

提 示 可选的 figcaption 必须与其他内容一起包含在 figure 里面，不能单独出现在其他位置。

提 示 figcaption 中的文本不必以 Figure3、Exhibit B 等字样开头，一段对内容的简短描述即可，就像图片的标题一样。

提 示 如果要在 figure 中包含 figcaption，它必须是 figure 的第一个或最后一个元素。

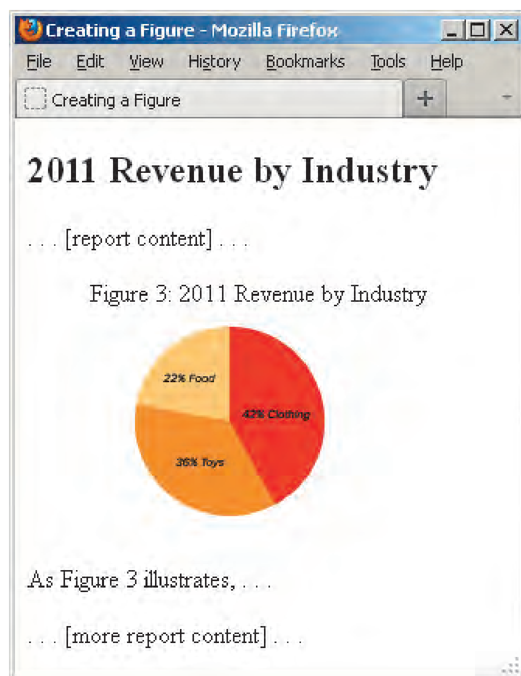


图 4.3.2 这个包含图表及其标题的 figure 出现在 article 文本中间。很容易使用 CSS 对该 figure 添加样式，例如为它添加边框，让文字环绕在它周围

4.4 指定时间

可以通过 time 元素标记一个准确的时间或日期，这是 HTML5 新增的元素。（关于日期体系的详细说明，参见本节末尾“理解 datetime 格式”。）

time 最常见的一种用法是表示 article 元素的发布日期。要表示发布日期，需要包含 pubdate 属性。包含 pubdate 属性的 time 元素表示其最近的祖先 article 元素的发布日期（参见图 4.4.1）。还可以用 time、datetime 和 pubdate 为 article 的读者评论添加时间戳（假定与 article 相关的评论都由嵌套在这个 article 中的 article 元素包着；参见 3.9 节“如何在 article 和 section 中作出选择”中的例 2）。

在使用 time 元素时，可以用多种方式表示时间（参见图 4.4.1 和图 4.4.3）。time 中可选的文本内容（即 <time> 文本 </time>）会出现在屏幕上，作为可选的、机器可读的 datetime 值的人工可读版本（参见图 4.4.2 和图 4.4.4）。

```
...
<body>

<article>
    <header>
        <h1>Cheetah and Gazelle Make Fast
        ➝ Friends</h1>
        <p><time datetime="2011-10-15"
        ➝ pubdate="pubdate">October 15,
        ➝ 2011</time></p>
    </header>

    ... [文章内容] ...
</article>

</body>
</html>
```

图 4.4.1 datetime 属性和 time 元素中的文本最好反映相同的日期，但它们在书写方式上可以不一样（更多例子见图 4.4.3）。这个 time 元素代表文章发布的日期，因为它包含了 pubdate 属性



图 4.4.2　article 的发布日期出现在标题下面。显示出来的是 time 元素的文本内容，而不是 datetime 值

指定准确时间和日期

(1) 输入 <time 开始 time 元素。

(2) 如果需要，输入 datetime="*time*"，其中 *time* 应使用合法的格式（参见本节末尾“理解 datetime 格式”）。

(3) 如果这个时间代表的是 article 或整个页面的发布日期，则输入 pubdate="pubdate" 或 pubdate。

(4) 输入 > 结束这个开始标记。

(5) 如果要让时间显示在浏览器中，则输入反映时间或日期的文本（关于允许的格式，参见第一条提示）。

(6) 输入 </time>。

提 示　如果忽略 datetime 属性，文本内容就必须是合法的日期或时间格式。也就是说，图 4.4.3 第一个例子不能写成 <p>The train arrives at <time>8:45 a.m.</time> and <time>4:20 p.m.</time> on <time>October 4th, 2012</time>.</p>。如果包含了 datetime，文本内容就可以按你希望的任何形式表示日期或时间了，就像图 4.4.3 中第二个和第三个例子那样。

提 示　不要在 time 中使用不准确的日期或时间，如“20 世纪中期”、“午夜过后”、“文艺复兴后期”或“上周早些时候”。

提 示　如果要在页面中显示 time 元素，应该在该元素中包含时间和日期的文本版本。如果没有包含，浏览器理应根据 datetime 值显示对应的文本，但在本书写作之际，浏览器对这一特性的支持程度还很低，参见图 4.4.4。

```
...
<body>

<p>The train arrives at <time>08:45</time>
→ and <time>16:20</time> on
→ <time>2012-04-10</time>.</p>

<p>We began our descent from the peak of
→ Everest on <time datetime="1952-06-12T11:
→ 05:00">June 12, 1952 at 11:05 a.m.
→ </time></p>

<p>They made their dinner reservation
→ for <time datetime="2011-09-20T18:
→ 30:00">tonight at 6:30</time>.</p>

<p>The record release party is on <time
→ datetime="2011-12-09"></time>.</p>

</body>
</html>
```

图 4.4.3　使用 time 元素有多种方式。最简单的形式（第一个例子）没有 datetime 属性。忽略 datetime 属性时，必须使用合法的格式表示日期和时间。头三个例子在 time 元素里都包含了时间或日期的文本，这些文字会显示在屏幕上（参见图 4.4.4）。建议始终包含这种人类可读的时间，因为当前的浏览器不会显示属性中的值（参见图 4.4.2）

提 示　如果使用带 pubdate 的 time 元素指示 article 的发布日期，通常将它放置在这个 article 的 header 或 footer 元素里，尽管这不是必需的。不管怎样，一定要把它嵌套在相关的 article 里面的某个位置。

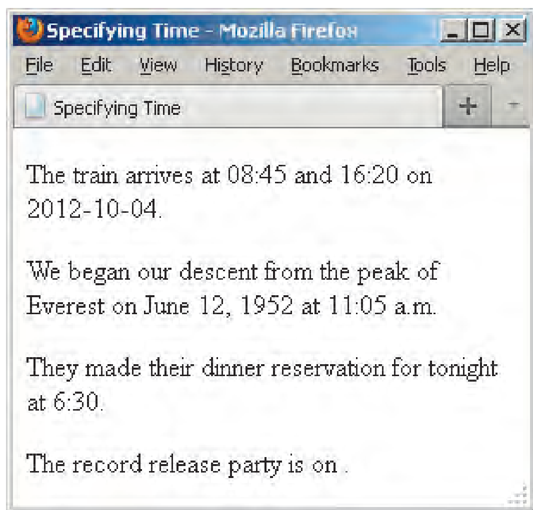


图 4.4.4　头三个段落会显示时间，最后一个不会（参见最后一条提示）

提 示　如果带 pubdate 属性的 time 元素的祖先中没有 article 元素，则代表整个页面的发布时间。

提 示　既可以用 <time **pubdate**></time>，也可以用 <time **pubdate="pubdate"**></time> 指定 pubdate。不过，一旦包含它，就必须提供 datetime 或时间的文本内容（参见图 4.4.1）。

提 示　datetime 属性的机器可读格式（参见本节末尾“理解 datetime 格式”）使 Web 应用之间可以同步日期和时间。不过在本书写作之际，还没有浏览器会显示 datetime 值（参见图 4.4.2 和图 4.4.4）。

提 示　不能在 time 元素中嵌套另一个 time 元素。

理解 datetime 格式

time 元素的时间是以 24 小时制为基础的，也可以通过相对 UTC（Coordinated Universal Time，全球标准时间）时差的方式来表示。datetime 属性应提供机器可读的日期和时间格式，其简化形式为：

YYYY-MM-DDThh:mm:ss

例如（当地时间）：

2011-11-03T17:19:10

表示“当地时间 2011 年 11 月 3 日下午 5 时 19 分 10 秒”。T 用于将日期（YYYY-MM-DD）和时间（hh:mm:ss）隔开。如果包含时间，秒是可选的。也可以使用 hh:mm.sss 格式提供时间的毫秒数。注意毫秒数之前的符号是一个点。

如果你愿意，也可以表示为世界时。在末尾加上字母 Z，就成了 UTC。

例如（世界时）：

2011-11-03T17:19:10Z

也可以通过相对 UTC 时差的方式表示时间。这时不写字母 Z，写上 -（减）或 +（加）及时差即可。

例如（含相对 UTC 时差的世界时）：

2011-11-03T17:19:10-03:30

4

表示“纽芬兰标准时（比 UTC 晚 3 个半小时）2011 年 11 月 3 日下午 5 时 19 分 10 秒”。UTC 时区列表参见 http://en.wikipedia.org/wiki/List_of_time_zones_by_UTC_offset。

如果确实要包含 datetime，也不必像图 4.4.3 中的示例那样提供时间的完整信息。从技术上来说，time 元素中的日期是以公历为基础的（公历是当前广泛使用的国际公认的日历体系）。因此，HTML5 不推荐使用该元素表示现行公历开始使用之前的日期（一般情况下这不会给你的内容带来影响，你只需要知道这一点就行了）。有很多关于这种限制的讨论，但这是一个复杂的主题。更多信息和实例参见 http://dev.w3.org/html5/spec-author-view/the-time-element.html；关于其中一些问题的详细说明，参见 www.quirksmode.org/blog/archives/2009/04/making_time_saf.html。

4.5　标记重要或强调的文本

strong 元素表示重要的文本，而 em 则表示强调。根据内容需要，这两个元素既可以单独使用，也可以一起使用，参见图 4.5.1 和图 4.5.2。

```
...
<body>

<p><strong>Warning: Do not approach the
➝ zombies <em>under any circumstances</em>.
➝ </strong> They may <em>look</em>
➝ friendly, but that's just because they want
➝ to eat your arm.</p>

</body>
</html>
```

图 4.5.1　第一个句子既有 strong 又有 em，而第二个句子只有 em。如果将 under any circumstances 改为 under any circumstances，它的重要程度会比 strong 包含的文本更高

1. 标记重要文本

(1) 输入 。

(2) 输入想标记为重要内容的文本。

(3) 输入 。

2. 强调文本

(1) 输入 。

(2) 输入想强调的文本。

(3) 输入 。

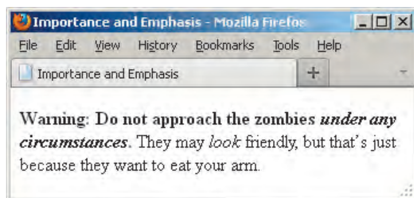


图 4.5.2　浏览器通常将 strong 文本以粗体显示，将 em 文本以斜体显示。如果 em 是 strong 的子元素（如图 4.5.1 中第一个句子），文本将同时以斜体和粗体显示

提 示　不要使用 b 元素代替 strong，也不要使用 i 元素代替 em。尽管它们在浏览器中显示的样式是一样的，但它们的含义却很不一样（参见本节末尾“HTML5 中重新定义的 b 和 i 元素”）。

提 示　可以在标记为 strong 的短语中再嵌套 strong 文本。如果这样做，作为另一个 strong 的子元素的 strong 文本的重要程度都会递增。这种规则对嵌套在另一个 em 里的 em 文本也适用。例如，在语句 `<p><strong>Remember that entries are <strong>due by November 17th</strong>.</strong></p>` 中，due by November 17th 要比其他 strong 文本更为重要。

提示　可以用 CSS 将任何文本变为粗体或斜体，也可以覆盖 strong 和 em 等元素的浏览器默认样式，参见图 4.5.2。更多细节参见 10.3 节和 10.4 节。

HTML5 中重新定义的 b 和 i 元素

HTML5 强调元素的语义，而非表现。b 和 i 元素是早期 HTML 遗留下来的产物，它们分别用于将文本变为粗体和斜体（那时 CSS 还未出现）。HTML 4 和 XHTML 1 当然抛弃了它们，因为它们本质上是用于表现的。当时的规范建议编码人员用 strong 替代 b，用 em 替代 i。不过，事实证明，em 和 strong 并不总是在语义上合适。为此，HTML5 重新定义了 b 和 i。

传统出版业里的某些排版规则在现有的 HTML 语义还找不到对应物，其中就包括用斜体表示某些科学名称（如马萨诸塞州州树 *Ulmus americana*）、具体的交通工具名称（如 *Orient Express*）及外来语（如 *joie de vivre*）。这些词语不是为了强调而加上斜体的，只是样式上的惯例。

为了应对这些情况，HTML5 没有创建一些新的语义化元素（进一步把事情搞复杂），而是采取了一种很实际的做法，直接利用现有元素：em 用于所有层次的强调，strong 用于表示重要性，而 b 和 i 则用于其他的情况。

这意味着，尽管 b 和 i 并不包含任何明显的语义，但读者仍能发现它们与周边文字的差别。而且你还可以通过 CSS 改变它们粗体或斜体的样式。HTML5 强调，b 和 i 应该是其他元素（如 strong、em、cite 等）都不适用时的最后选择。

b 元素简介

HTML5 将 b 重新定义为：

> b 元素表示出于实用的目的提醒读者注意的一块文字，不传达任何额外的重要性，也不表示其他的语态和语气，用于如文档摘要里的关键词、评论里的产品名、基于文本的交互式软件中指示操作的文字、文章导语等。

例如：

```
<p>The <b>XR-5</b>, also dubbed the <b>Extreme Robot 5</b>, is the best robot we've ever
➝ tested.</p>
```

b 元素默认显示为粗体。

i 元素简介

HTML5 将 i 重新定义为：

> i 元素表示一块不同于其他文字的文字，具有不同的语态或语气，或其他不同于常规之处，用于如分类名称、技术术语、外语里的惯用语、思想、西方文字中的船舶名称等。

例如：

```
<p>The <i lang="la">Ulmus americana</i> is the Massachusetts state tree.</p>
<p>The <i>Orient Express</i> began service in 1883.<p>
<p>The couple exhibited a <i lang="fr">joie de vivre</i> that was infectious.<p>
```

i 元素默认显示为斜体。

4.6 指明引用或参考

使用 cite 元素可以指明对某内容源的引用或参考。例如，戏剧、脚本或图书的标题，歌曲、电影、照片或雕塑的名称，演唱会或音乐会，规范、报纸或法律文件，等等，参见图 4.6.1 和图 4.6.2。

```
...

<p>He listened to <cite>Abbey Road</cite>
→ while watching <cite>A Hard Day's Night
→ </cite> and reading <cite>The Beatles
→ Anthology</cite>.

<p>When he went to The Louvre, he learned
→ that <cite>Mona Lisa</cite> is also known
→ as <cite lang="it">La Gioconda</cite>.</
p>

...
```

图 4.6.1　cite 元素适用于标记艺术作品、音乐、电影和图书等的标题

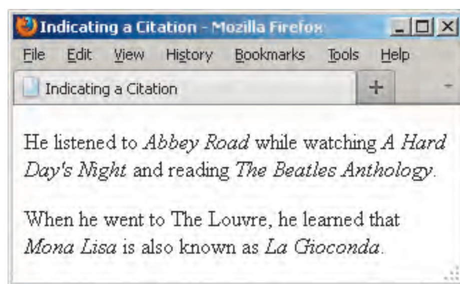


图 4.6.2　cite 元素默认以斜体呈现

引用参考的步骤

(1) 输入 <cite>。

(2) 输入参考的名称。

(3) 输入 </cite>。

提 示　对于要从引用来源中引述内容的情况，使用 blockquote 或 q 元素标记引述的文本（参见 4.7 节）。要弄清楚的是，cite 只用于参考源本身，而不是从中引述的内容。

HTML5 与使用 cite 元素表示名称

HTML5 声明，不应使用 cite 作为对人名的引用（这是开发社区众多意见分歧中的一个），但 HTML 以前的版本允许这样做，而且很多设计和开发人员仍在这样做。

HTML 4 的规范有以下例子（已将元素名称由大写字母改为小写）：

```
As <cite>Harry S. Truman</cite> said,
<q lang="en-us">The buck stops here.</q>
```

除了这些例子，有的网站经常用 cite 标记在博客和文章中发表评论的访问者的名字（WordPress 默认模板就是这样做的）。

很多开发人员表示他们将继续对与页面中的引文有关的名称使用 cite，因为 HTML5 没有提供其他他们认为可接受的元素（即 span 和 b 元素）。Jeremy Keith 提供了一个很好的例子，见 http://24ways.org/2009/incite-a-riot/。

4.7 引述文本

有两个特殊的元素用以标记引述的文本。blockquote 元素表示单独存在的引述（通常更长，但不一定这样），参见图 4.7.1，默认显示在新的一行，参见图 4.7.2。而 q 元素则用于短的引述，像那些句子里面的引述，参见图 4.7.3。

```
...
<body>

<p>He enjoyed this selection from <cite>The
→ Adventures of Huckleberry Finn</cite> by
→ Mark Twain:</p>

<blockquote cite="http://www.marktwain
→ books.edu/the-adventures-of-huckleberry
→ -finn/">
    <p>We said there warn't no home like a
    → raft, after all. Other places do seem
    → so cramped up and smothery, but a
    → raft don't. You feel mighty free and
    → easy and comfortable on a raft.</p>
</blockquote>

<p>It reminded him of his own youth exploring
→ the county by river.</p>

</body>
</html>
```

图 4.7.1 根据需要，blockquote 可长可短。可以包含 cite 属性（不要与第一段中出现的 cite 元素混淆）提供引述文本的位置。不过，浏览器不会显示 cite 属性中的内容，参见图 4.7.2（要了解推荐的做法，参见第二条提示）

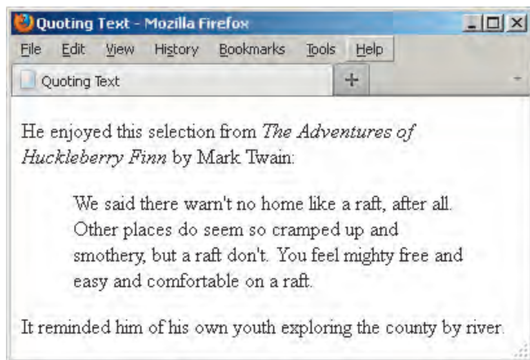


图 4.7.2 浏览器默认对 blockquote 文本进行缩进。截至目前，还没有浏览器会显示 cite 属性的值（要了解推荐的做法，参见第二条提示）。但所有的浏览器都支持 cite 元素，通常对其中的文本以斜体显示，如图所示。所有这些默认样式都可以被 CSS 覆盖

```
...
<body>

<p>And then she said, <q>Have you read
→ Barbara Kingsolver's <cite>High Tide in
→ Tucson</cite>? It's inspiring.</q></p>

<p>She tried again, this time in French:
→ <q lang="fr">Avez-vous lu le livre
→ <cite>High Tide in Tucson</cite> de
→ Kingsolver? C'est inspirational.</q></p>

</body>
</html>
```

图 4.7.3 如果引述文本的语言与页面默认语言（通过 html 元素的 lang 属性指定）不同，就在 q 元素中加上 lang 属性

浏览器应对 q 元素中的文本自动加上特定语言的引号，但 Internet Explorer 直到 IE8 才开始支持这一特性。还有的浏览器在处理嵌套的引述时存在一些问题。一定要阅读本节提示，了解 q 元素的替代方法。

1. 引述块级文本的步骤

(1) 输入 <blockquote 开始一个块级引述。

(2) 如果需要，输入 cite="*url*"，其中 *url* 为引述来源的地址。

(3) 输入 > 以结束开始标记。

(4) 输入希望引述的文本，并用段落等适当的元素包围。

(5) 输入 </blockquote>。

2. 引述行内文本的步骤

(1) 输入 <q 开始引述字词或短语。

(2) 如果需要，输入 cite="*url*"，其中 *url* 为引述来源的地址。

(3) 如果引述内容的语言与页面默认语言（通过 html 元素的 lang 属性指定）不同，输入 lang="*xx*"，其中 *xx* 是引述内容语言的两个字母的代码。这个代码用于判断需要使用的引号的类型（英语为 “”，而很多欧洲语

言则为 «»），但浏览器对引号的呈现方式可能不同。

(4) 输入 > 以结束开始标记。

(5) 输入要引述的文本。

(6) 输入 </q>。

提示　尽管这样做也是允许的，但应避免将文本直接放在 blockquote 开始和结束标记之间。应该将文本放在 blockquote 中 p 或其他语义上适合的元素中。

提示　可以对 blockquote 和 q 使用可选的 cite 属性，提供引述内容来源的 URL。不幸的是，浏览器通常不会将 cite 的 URL 呈现给用户，参见图 4.7.2，因此这个属性本身不是特别有用。因此，建议在内容中使用链接（a 元素）重复这个 URL，让访问者可以查看来源。也可以使用 JavaScript 将 cite 的值暴露出来，但这样做的效果稍差一些。

提示　blockquote 元素是 HTML5 的区块根，这意味着它可以有 h1 ～ h6 标题（及其自身的大纲），但文档大纲不会包含这些标题。这与分块内容是不一样的，参见 3.4 节。

提示　q元素引用的内容不能跨越不同的段落。

提示　不要仅仅因为需要在字词两端添加引号就使用 q。例如，<p>Every time I hear the word **<q>soy</q>**, I jump for joy.</p> 是不正确的，因为 soy 并不是对某参考源的引用。

提示　blockquote 和 q 元素可以嵌套。例如，<p>The short story began, **<q>When she was a child, she would say, <q>Howdy, stranger!</q> to everyone she passed.</q>**</p>。嵌套的 q 元素应该自动加上正确的引用（在英语中外面的是双引号，里面的是单引号），但浏览器的支持程度并不相同。由于内外引号在不同语言中的处理方式是不一样的，因此要根据需要在 q 元素中加上 lang 属性（参见图 4.7.3 和图 4.7.4）。

提示　由于 q 元素的跨浏览器问题（参见图 4.7.4），很多（可能是大多数）开发人员避免使用 q 元素，而是选择直接输入正确的引号或使用字符实体。Oli Studholme 在 HTML5 Doctor 上发表的文章 “Quoting and citing with <blockquote>, <q>, <cite>, and the cite attribute” 深入分析了这一话题，还提供了一些为 q 元素添加引号样式的方法和相关的浏览器支持信息（http://html5doctor.com/blockquote-q-cite/）。

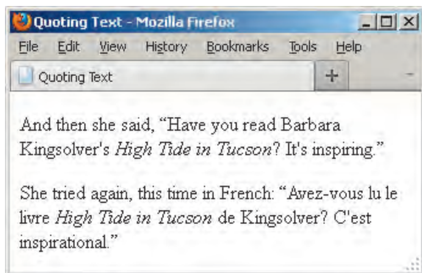


图 4.7.4　浏览器应该自动在 q 元素周围加上双引号（此外，应在嵌套的 q 元素周围加上单引号）。如图所示，Firefox 是这样做的，但并非所有的浏览器都支持（例如旧版本的 Internet Explorer）

4.8　突出显示文本

我们都在某些时候用过荧光笔——也许是在复习考试，抑或是在审查合同。无论是哪种情况，荧光笔标记的都是一些与任务有关的关键字词。

HTML5 通过新的 mark 元素复制这一过程。试着将 mark 想成荧光笔的语义化对照物。换句话说，重要的是对特定的词语进行标注，与它们如何显示无关。可以用 CSS 对 mark 元素里的文字应用样式（或者不应用任何样式），但应仅在合适的情况下使用该元素。

```
…
<body>

<p>So, I went back and read the instructions
→ myself to see what I'd done wrong. They
→ said:</p>

<blockquote>
    <p>Remove the tray from the box. Pierce
    → the overwrap several times with a
    → fork and cook on High for <mark>15
    → minutes</mark>, rotating it half way
    → through.</p>
</blockquote>

<p>I thought he'd told me <em>fifty</em>. No
→ wonder it exploded in my microwave.</p>

</body>
</html>
```

图 4.8.1　尽管 mark 最常见的使用场合是在搜索结果页面，这里给出了另一种合理的用法。15 minutes 一词在包装上的说明中没有被突出显示，但这个 HTML 的作者却使用 mark 将该词进行突出显示，支撑所叙述的故事。浏览器对 mark 文本的呈现样式存在差异（参见图 4.8.2）

无论何时使用 mark，该元素总是用于提起读者对特定文本片段的注意。下面是一些应用的例子。

- 对搜索结果页面或文章中的搜索词进行突出显示。人们讨论 mark 时，这是最常见的应用场景。假设使用网站的搜索功能查找“solar panels”。搜索结果或每个结果文章可以使用 <mark>solar panels</mark>，从而在整篇文字中突出显示该词。
- 对于某段引述，如果作者在原来的格式中没有对其进行突出显示，可以用 mark 引起对它的注意（参见图 4.8.1 和图 4.8.2）。
- 引起对代码片段的注意（参见图 4.8.3 和图 4.8.4）。

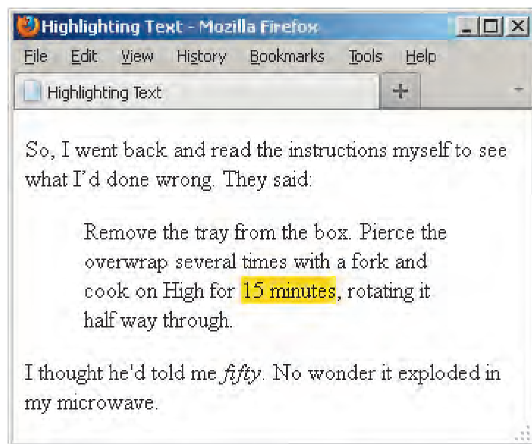


图 4.8.2　对 mark 原生支持的浏览器将对该元素的文字默认加上黄色背景。旧的浏览器不会这样做，但通过一条简单的样式表规则就可以让它们实现这种效果（参见提示）

```
…
<body>

<p>It's bad practice to use a class name that
→ describes how an element should look, such
→ as the highlighted portion of CSS below:
<pre>
    <code>
    <mark>.greenText</mark> {
       color: green;
    }
    </code>
</pre>

</body>
</html>
```

图 4.8.3　这个例子使用 mark 引起对代码片段的注意

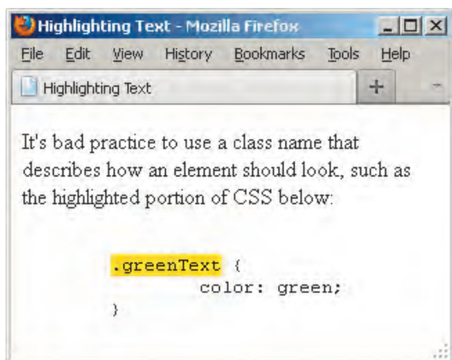


图 4.8.4　由 mark 标记的代码被突出显示了

突出显示文本的步骤

(1) 输入 <mark>。

(2) 输入希望引起注意的字词。

(3) 输入 </mark>。

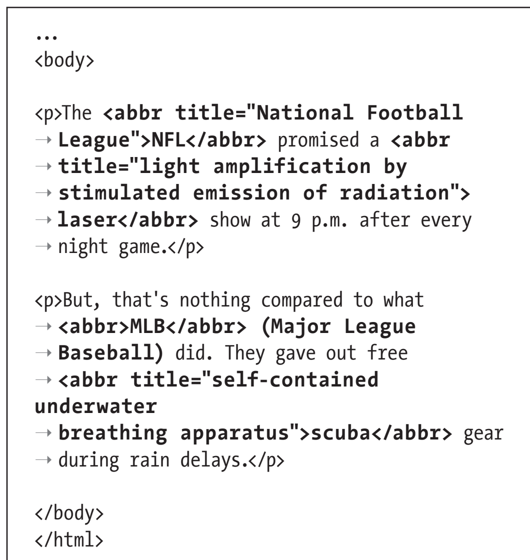
提示　mark 元素与 em（表示强调）或 strong（表示重要）都不相同。这两个元素都在本章前面讲过了。

提示　由于 mark 是 HTML5 的新元素，因此旧的浏览器不会默认加上黄色背景（参见图 4.8.2 和图 4.8.4）。可以在样式表中加上 **mark { background-color: yellow; }** 让这些浏览器实现同样的效果。

提示　不要仅仅因为要给文字加上背景颜色或其他视觉上的考虑而使用 mark。如果你只是要给一块文字添加样式，又没有合适的语义化元素，就可以使用 span 元素（本章将讲到），并用 CSS 添加样式。

4.9　解释缩写词

缩写词很常见，如 Jr.、M.D.，甚至 good ol' HTML。可以使用 abbr 元素标记缩写词并解释其含义（参见图 4.9.1、图 4.9.2 和图 4.9.3）。不必对每个缩写词使用 abbr，只在需要帮助访问者了解该词含义的时候使用。


```
...
<body>

<p>The <abbr title="National Football
→ League">NFL</abbr> promised a <abbr
→ title="light amplification by
→ stimulated emission of radiation">
→ laser</abbr> show at 9 p.m. after every
→ night game.</p>

<p>But, that's nothing compared to what
→ <abbr>MLB</abbr> (Major League
→ Baseball) did. They gave out free
→ <abbr title="self-contained
underwater
→ breathing apparatus">scuba</abbr> gear
→ during rain delays.</p>

</body>
</html>
```

图 4.9.1　使用可选的 title 属性提供缩写词的全称。另外，也可以将全称放在缩写词后面的括号里（这样做或许更好）。还可以同时使用这两种方式，并使用一致的全称。对于像 laser（激光）、scruba（水肺潜水）这样的词汇，大多数人都很熟悉了，因此对它们使用 abbr 并提供 title 其实并无必要，这里使用它们是出于演示目的

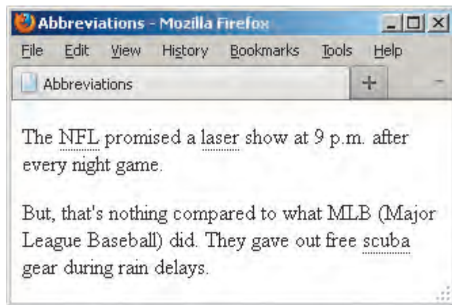


图 4.9.2　当缩写词有 title 属性时，Firefox 和 Opera 会添加虚线下划线以引起注意。（可以通过 CSS 让其他浏览器也这样做；参见提示。）对除 IE6 以外的所有浏览器，当访问者将鼠标移至 abbr 上，该元素 title 属性的内容就会显示在一个提示框里

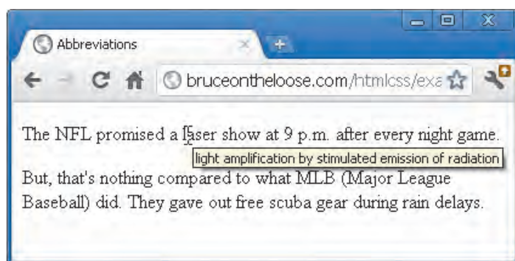


图 4.9.3 Chrome 和一些其他的浏览器会将缩写词的 title 显示在提示框里，但它们不会让缩写词本身有显示上的区别，除非你自行添加一些 CSS

解释缩写词的步骤

(1) 输入 <abbr。

(2) 作为可选的一步，输入 title="*expansion*"，其中 *expansion* 是缩写词的全称。

(3) 输入 >。

(4) 然后输入缩写词本身。

(5) 最后输入 </abbr> 结束标记。

(6) 作为可选的一步，输入一个空格和 (*expansion*)，其中 *expansion* 是缩写词的全称。

提示 通常，仅在缩写词第一次出现在屏幕上时给出其全称（通过 title 或括号的方式），参见图 4.9.1。

提示 用括号提供缩写词的全称是最直接的解释缩写词的方式，能让尽可能多的访问者看到这些内容，参见图 4.9.1。例如，使用智能手机和平板电脑等触摸屏设备的用户可能无法移到 abbr 元素上查看 title 的提示框。因此，如果要提供缩写词的全称，应该尽量将它放在括号里。

提示 如果使用复数形式的缩写词，全称也要使用复数形式。

提示 作为对用户的视觉提示，Firefox 和 Opera 等浏览器会对带 title 的 abbr 文字使用虚线下划线，参见图 4.9.2。如果要在所有的浏览器（IE6 除外）中复制这种样式，可以在样式表中加上这条语句：**abbr[title] { border-bottom: 1px dotted #000; }**。无论 abbr 是否添加了下划线样式，浏览器都会将 title 属性内容以提示框的形式显示出来，参见图 4.9.3。

提示 如果在 Internet Explorer 7 中看不到 abbr 有虚线下划线，试着为其父元素的 CSS 添加 line-height 属性（参见第 10 章）。

提示 IE6 不支持 abbr，因此既没有虚线下划线也没有提示框，只有文字本身。如果确实希望在 IE6 中为 abbr 应用样式，可以在 CSS 之前将 document.createElement('abbr'); 放在针对 IE6 的 JavaScript 文件里。要我说，不如略过这一办法，把 IE6 当作一个例外。（关于在 IE8 及旧版本中使用 document.createElement 让元素看起来像 HTML5 中添加样式的新元素，参见第 11 章。）

提示 在 HTML5 之前有 acronym（首字母缩写词）元素，但设计和开发人员常常分不清楚缩写词和首字母缩写词，因此 HTML5 废除了 acronym 元素，让 abbr 适用于所有的场合。

4.10 定义术语

dfn 元素对文档中首次定义术语的地方进行标记。并不需要用它标记术语的后续使用。仅用 dfn 包围要定义的术语，而不是包围定义，参见图 4.10.1。

```
…
<body>

<p>The contestant was asked to spell
→ "pleonasm." She requested the definition
→ and was told that <dfn>pleonasm</dfn>
→ means "a redundant word or expression"
→ (Ref: <cite><a href=" http://dictionary.
→ reference.com/browse/pleonasm" rel=
→ "external">dictionary.com</a></cite>).</p>

</body>
</html>
```

图 4.10.1 注意，在这个例子中，尽管 pleonasm 出现了两次，但只对第二个用 dfn 进行了标记，因为那时才定义这个术语（即定义实例）。类似地，如果我在文档后面用到 pleonasm，也不会用 dfn 标记，因为我已经定义过它了。默认情况下，浏览器会为 dfn 文本加上与普通文本不同的样式，参见图 4.10.2。同时，不必在每次使用 dfn 时都使用 cite 元素，该元素只在引用参考源的时候使用

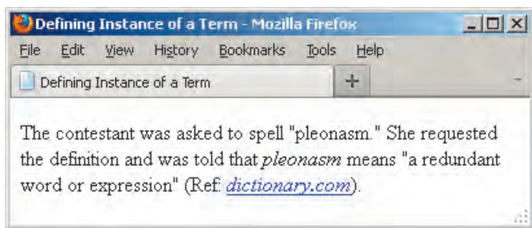


图 4.10.2 一些浏览器（如这里的 Firefox）默认以斜体显示 dfn 元素，与 cite 一样。不过，基于 Webkit 的浏览器（如 Safari 和 Chrome）不会这样做。为了让它们统一，可以在样式表中加入 dfn { font-style: italic; }（参见第 8 章和第 10 章）

将与定义有关的 dfn 放在什么位置是一个重要的问题。HTML5 规定：“作为距离 dfn 元素最近的祖先区块的段落、描述列表组或区块，除了包含 dfn 元素以外，还必须包括该术语的定义。”这意味着 dfn 及术语的定义应该是彼此相邻的。图 4.10.1 和第三个提示中的例子都是这样，dfn 及术语的定义位于同一个段落。

标记术语的定义实例

(1) 输入 <dfn>。

(2) 输入要定义的术语。

(3) 输入 </dfn>。

提 示 还可以在定义列表（dl 元素）中使用 dfn，参见第 15 章。

提 示 如果要将用户引向术语的定义实例，可以给 dfn 加个 id，然后让站点其他位置的链接指向它。

提 示 dfn 可以在适当的情况下包住其他的短语元素，如 abbr。例如，<p>A **<dfn><abbr title="Junior">Jr.</abbr></dfn>** is a son with the same full name as his father.</p>。

提 示 HTML5 称，如果在 dfn 中添加可选的 title 属性，其值应与 dfn 术语一致。正如在上一条提示中看到的，如果只在 dfn 里嵌套一个单独的 abbr，dfn 本身没有文本结点，那么可选的 title 只能出现在 abbr 里。

4.11 创建上标和下标

比主体文本稍高或稍低的字母或数字分别称为上标和下标，其表示参见图 4.11.1。HTML 包含用来定义这两种文本的元素。常见的上标包括商标符号、指数和脚注编号等，参见图 4.11.2。常见的下标包括化学符号等。

创建上标和下标的步骤

(1) 输入 <sub> 创建下标，或输入 <sup> 创建上标。

(2) 输入要出现在下标或上标里的字符或符号。

(3) 根据第 (1) 步中使用的元素，输入 `</sub>` 或 `</sup>` 结束该元素。

```
...
<body>

<article>
    <h1>Famous Catalans</h1>
    <p>When I was in the sixth grade, I
    → played the cello. There was a
    → teacher at school who always used
    → to ask me if I knew who "Pablo
    → Casals" was. I didn't at the time
    → (although I had met Rostropovich once
    → at a concert). Actually, Pablo Casals'
    → real name was <i>Pau</i> Casals, Pau
    → being the Catalan equivalent of Pablo
    → <a href="#footnote-1" title="Read
    → footnote"><sup>1</sup></a>.</p>

    <p>In addition to being an amazing
    → cellist, Pau Casals is remembered in
    → this country for his empassioned
    → speech against nuclear proliferation
    → at the United Nations <a href=
    → "#footnote-2" title="Read
    → footnote"><sup>2</sup></a> which
    → he began by saying "I am a Catalan.
    → Catalonia is an oppressed nation."</p>

    <footer>
       <p><sup>1</sup>It means Paul in
       → English.</p>
       <p><sup>2</sup>In 1963, I believe.</p>
    </footer>
</article>

</body>
</html>
```

图 4.11.1 sup 元素的一种用法就是表示脚注编号。根据从属关系，我将脚注放在 article 的 footer 里，而不是整个页面的 footer 里。我还为文章中每个脚注编号创建了链接，指向 footer 里对应的脚注，从而让访问者更容易找到它们。同时，注意链接里的 title 属性也提供了一些提示

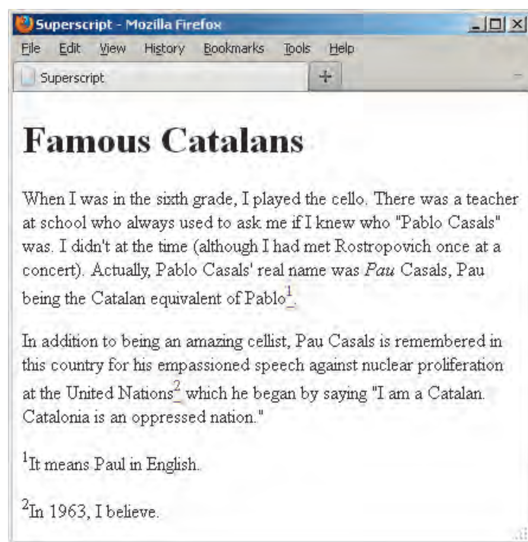


图 4.11.2 不幸的是，sub 和 sup 元素会扰乱行间距。注意第一段第 4 行和第 5 行之间的距离以及第二段第 2 行和第 3 行之间的距离比其他行间距要大一些。不过，可以使用一些 CSS 解决这个问题。关于解决的办法，参见本节末尾的“修复使用 sub 和 sup 时的行间距问题”。还可以对带链接的上标进行适当的处理，让下划线不至于离上标文字太远

提 示 大多数浏览器会自动将上标或下标文字的字号减少几磅。

提 示 上标是对某些外语缩写词进行格式化的理想方式，例如，法语中用 Mlle 表示 Mademoiselle（小姐），西班牙语中用 3^a 表示 tercera（第三）。此外，一些数字形式也要用到上标，如 2nd、5th。

提 示 下标适用于化学分子式（如 H_2O）。例如，`<p>I'm parched. Could I please have a glass of H<sub>2</sub>O?</p>`。

提 示 上标和下标字符会轻微地扰乱行与行之间均匀的间距，参见图 4.11.2。解决的办法参见“修复使用 sub 和 sup 时的行间距问题”。

修复使用 sub 和 sup 时的行间距问题

sub 和 sup 元素会轻微地增大行高，参见图 4.11.2。幸好，用一点 CSS 就可以修复这个问题。

下面的代码来自 Nicolas Gallagher 和 Jonathan Neal 出色的 normalize.css（http://necolas.github.com/normalize.css/）。下面的方法并不是他们发明的，他们借用了 https://gist.github.com/413930 里的代码并去掉了注释。上面这个 GitHub 的链接里包含了对这一段 CSS 的详细解释，建议你去看一看。同时，你可以在自己的项目里使用 normalize.css，推荐你下载这个文件。该文件可以帮你建立一个跨浏览器的统一基准样式表，其文档也很详尽（参见 11.4 节）。

```
/*
* 在所有浏览器中防止 sub 和 sup 影响 line-height
* gist.github.com/413930
*/
sub,
sup {
    font-size: 75%;
    line-height: 0;
    position: relative;
    vertical-align: baseline;
}

sup {
    top: -0.5em;
}

sub {
    bottom: -0.25em;
}
```

你可能还需要根据内容的字号对这个 CSS 做一些调整，使各行行高保持一致，但上面的代码至少已经为你建立了很好的起点。关于如何创建样式表并将 CSS 添加到站点里，参见第 8 章。

4.12　标注编辑和不再准确的文本

有时可能需要将在前一个版本之后对页面内容的编辑标出来，或者对不再准确、不再相关的文本进行标记。有两种用于标注编辑的元素：代表添加内容的 ins 元素和标记已删除内容的 del 元素（参见图 4.12.1 ～图 4.12.4）。这两个元素既可以单独使用，也可以一起使用。

同时，s 元素用以标注不再准确或不再相关的内容（一般不用于标注编辑），参见图 4.12.5 和图 4.12.6。

1. 标记新插入文本

(1) 输入 <ins>。

(2) 输入新内容。

(3) 输入 </ins>。

2. 标记已删除文本

(1) 将光标放在待标记为已删除的文本或元素之前。

(2) 输入 <del>。

(3) 将光标放在待标记为已删除的文本或元素之后。

(4) 输入 </del>。

3. 标记不再准确或不再相关的文本

(1) 将光标放在希望标记为不再准确或不再相关文本的前面。

(2) 输入 <s>。

(3) 将光标放在希望标记的文本的后面。

(4) 输入 </s>。

```
...
<body>

<h1>Charitable Gifts Wishlist</h1>

<p>Please consider donating one or more
→ of the following items to the village's
→ community center:</p>

<ul>
    <li><del>2 desks</del></li>
    <li>1 chalkboard</li>
    <li><del>4 <abbr>OLPC</abbr> (One
    → Laptop Per Child) XO laptops
    → </del></li>
    <li><ins>1 bicycle</ins></li>
</ul>

</body>
</html>
```

图 4.12.1　在这个礼品清单上一次发布之后，又增加了一个条目（bicycle），同时根据 del 元素的标注，移除了一些已购买的条目。使用 ins 的时候不一定要使用 del，反之亦然。浏览器通常会让它们看起来与普通文本不一样，参见图 4.12.2

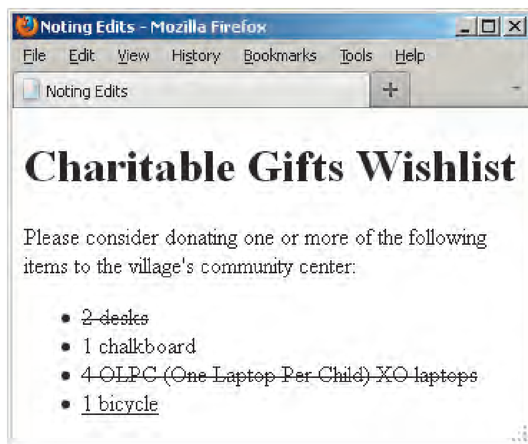


图 4.12.2　浏览器通常对已删除的文本加上删除线，对插入的文本加上下划线。可以用 CSS 修改这些样式

4

```
...
<body>

<h1>Charitable Gifts Wishlist</h1>

<del>
    <p>Please consider donating one or more of the following items to the village's community
    → center:</p>
</del>

<ins>
    <p>Please note that all gifts have been purchased.</p>
    <p>Thank you <em>so much</em> for your generous donations!</p>
</ins>

<del>
    <ul>
        <li><del>2 desks</del></li>
        <li>1 chalkboard</li>
        <li><del>4 <abbr>OLPC</abbr> (One Laptop Per Child) XO laptops</del></li>
        <li><ins>1 bicycle</ins></li>
    </ul>
</del>

</body>
</html>
```

图 4.12.3　del 和 ins 是少有的既可以包围短语内容（HTML5 之前称“行内元素”）又可以包围块级内容的元素，如下面的代码所示。不过，浏览器对它们的默认显示会有区别（参见图 4.12.4）



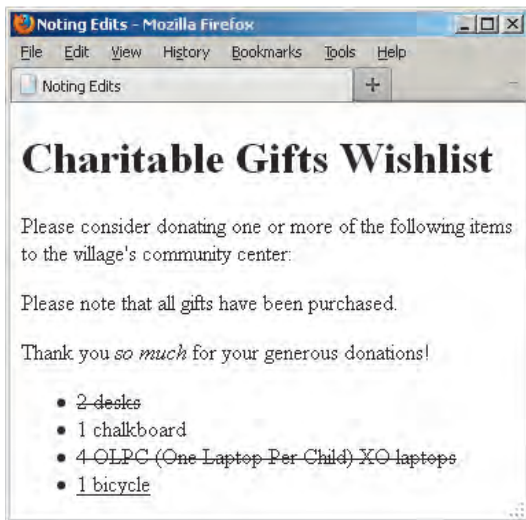


图 4.12.4　对于包围块级内容的 del 和 ins，大多数浏览器的默认显示样式与预期一致，如 Chrome（上图）。也就是说，它们反映整块内容都是删除的内容或插入的内容。截至本书写作之际，Firefox 不会这样做，它们只对包含在其他元素中的 del 和 ins 短语文本添加删除线或下划线。关于如何修正这一问题，参见本节末尾的“让 del 和 ins 的显示一致”

```
...
<body>

<h1>Today's Showtimes</h1>

<p>Tickets are available for the following
➝ times today:</p>

<ol>
    <li><ins>2 p.m. (this show just added!)
    ➝ </ins></li>
    <li><s>5 p.m.</s> SOLD OUT</li>
    <li><s>8:30 p.m.</s> SOLD OUT</li>
</ol>

</body>
</html>
```

图 4.12.5　这个例子展示了一个关于演出时间的有序列表（ol 元素）。与剩余票数不再相关的时段都用 s 元素进行了标记。可以对任何短语使用 s，而不仅限于列表项（li 元素）里的文本。不过，不要像 del 和 ins 那样用 s 标记整个段落或其他块级元素

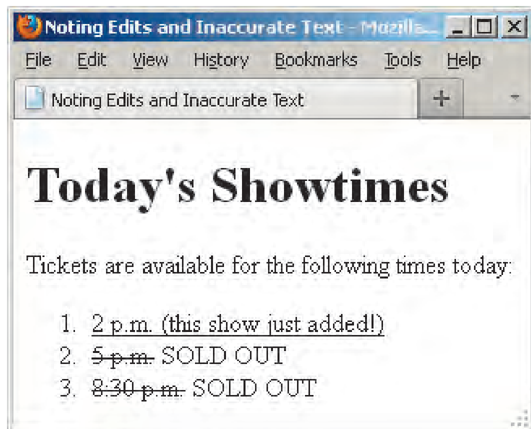


图 4.12.6　默认情况下，浏览器会对 s 元素添加删除线

提示 del 和 ins 都支持两个属性：cite 和 datetime。cite 属性（区别于 cite 元素）用于提供一个 URL，指向说明编辑原因的页面。例如，<ins **cite="http://www.movienews.com/ticket-demandhigh.html"**>2 p.m. (this show just added!)</ins>。datetime 属性提供编辑的时间。（关于 datetime 可接受的格式，参见 4.4 节。）浏览器不会将这两个属性的值显示出来，因此它们的使用并不广泛。不过，应该尽量包含它们，从而为内容提供一些背景信息。它们的值可以通过 JavaScript 或分析页面的程序提取出来。

提示 在任何需要向访问者展示内容变化情况的时候，都可以使用 del 和 ins。例如，经常可以看见一些 Web 设计与开发教程使用它们表示初次发布以后新更新的信息，这样可以保持原始信息的完整性。博客、新闻网站等也可以这样做。

提示 用 ins 标记的文本通常会显示一条下划线，参见图 4.12.2。由于链接通常也以下划线表示（即使你的网站不是这样，其他很多网站都是这样），这可能会让访问者感到困惑。可以使用样式表改变插入的段落（或链接）的显示（参见第 10 章）。

提示 用 del 标记的文本通常会显示一条删除线，参见图 4.12.2。为什么不直接清除这些文字呢？这取决于内容的上下文。加上删除线以后，用户就很容易看出修改了什么。（同时，屏幕阅读器可以读出被删除的内容，不过目前对这一特性的支持还不充分。）

提示 仅在有语义价值的时候使用 del、ins 和 s。如果只是出于装饰的原因要给文字添加下划线或删除线，可以用 CSS 实现这些效果（参见 10.16 节）。

提示 HTML5 指出："s 元素不适用于指出文档的编辑；要标记文档中一块已移除的文本，应使用 del 元素。"有时，这之间的差异是很微妙的，只能由你决定哪种选择更符合内容的语义。

让 del 和 ins 的显示一致

浏览器对块级 del 和 ins 内容的显示并不一致。大多数与预期一致，对 del 嵌套的内容添加删除线，对 ins 嵌套的内容添加下划线。不过，至少 Firefox 并不是这样做的，参见图 4.12.4。

可以通过下面的 CSS 规则修复这一问题（* 代表 del 和 ins 里面的所有元素均以下面的规则处理）：

```
del * {
    text-decoration: line-through;
}

ins * {
    text-decoration: underline;
}
```

如果你不了解如何将这些代码加到样式表，请参阅第 8 章。

4.13 标记代码

如果你的内容包含代码示例、文件名或程序名，就可以使用 code 元素（参见图 4.13.1 和图 4.13.2）。要显示单独的一块代码（位于句子以外），可以用 pre 元素包住 code 元素以维持其格式（具体示例参见 4.14 节）。

4

```
...
<body>

<p>The <code>showPhoto()</code> function
→ displays the full-size photo of the
→ thumbnail in our <code>&lt;ul id=
→ "thumbnail"&gt;</code> carousel list.</p>

<p>This CSS shorthand example applies a
→ margin to all sides of paragraphs:
→ <code>p { margin: 20px; }</code>. Take
→ a look at <code>base.css</code> to see
→ more examples.</p>

</body>
</html>
```

图 4.13.1　code 元素表示其中的文本是代码或文件名。默认情况下，它以等宽字体显示（参见图 4.13.2）。如果你的代码需要显示 < 或 >，应分别使用 < 和 >。这里第二个 code 的例子演示了这一点。如果真的用了 < 和 >，浏览器会将这些代码当做 HTML 元素处理，而不是当做文本处理

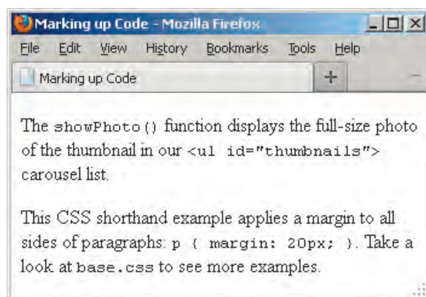


图 4.13.2　code 元素的文本使用等宽字体，使之看起来像代码

标记代码或文件名的步骤

(1) 输入 <code>。

(2) 输入代码或文件名。

(3) 输入 </code>。

提示　可以通过 CSS 改变 code（参见图 4.13.2）默认的等宽字体样式（参见第 10 章）。

其他计算机相关元素：kbd、samp 和 var

kbd、samp 和 var 元素极少使用，不过你可能会在内容中用到它们。下面对它们作简要说明。

- ❑ **kbd 元素**

  使用 kbd 标记用户输入指示。

```
<p>To log into the demo:</p>
<ol>
    <li>Type <kbd>tryDemo</kbd> in the User Name field</li>
    <li><kbd>TAB</kbd> to the Password field and type <kbd>demoPass</kbd></li>
    <li>Hit <kbd>RETURN</kbd> or <kbd>ENTER</kbd></li>
</ol>
```

  与 code 一样，kbd 默认以等宽字体显示。

- ❑ **samp 元素**

  samp 元素用于指示程序或系统的示例输出。

```
<p>Once the payment went through, the site returned a message reading,
→ <samp>Thanks for your order!</samp></p>
```

  samp 也默认以等宽字体显示。

- ❑ **var 元素**

  var 元素表示变量或占位符的值。

```
<p>Einstein is best known for <var>E</var>=<var>m</var><var>c</var><sup>2</sup>.
→ </p>
```

var 也可以作为内容中占位符的值，例如，在填词游戏的答题纸上可以放入 **<var>adjec-tive</var>**，**<var>verb</var>**。

var 默认以斜体显示。

需要注意的是，可以在 HTML5 页面中使用 math 等 MathML 元素表示高级的数学相关的标记。更多信息参见 http://dev.w3.org/html5/spec-author-view/mathml.html。

4.14 使用预格式化的文本

通常，浏览器会将所有额外的回车和空格压缩，并根据窗口的大小自动换行。预格式化的文本可以保持文本固有的换行和空格。它是计算机代码示例的理想元素，参见图 4.14.1，不过你也可以将它用于文本（比如，ASCII 艺术[①]）。

```
...
<body>

<p>Add this to your style sheet if you want
→ to display a dotted border underneath the
→ <code>abbr</code> element whenever it has
→ a <code>title</code> attribute.</p>

<pre>
    <code>
    abbr[title] {
       border-bottom: 1px dotted #000;
    }
    </code>
</pre>

</body>
</html>
```

图 4.14.1 对于包含重要的空格和换行的文本（如这里显示的 CSS 代码），pre 元素是非常适合的。同时要注意 code 元素的使用，该元素可以标记 pre 外面的代码块或与代码有关的文本（更多细节参见 4.13 节）

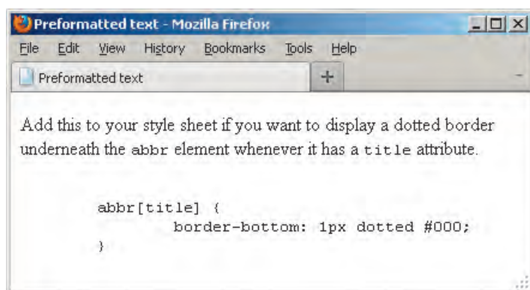


图 4.14.2 注意 pre 内容里的缩进和换行都被保留了

使用预格式化文本的步骤

(1) 输入 <pre>。

(2) 输入或复制希望以原样显示的文本，包括所需要的空格、回车和换行。除了代码以外，不要用任何 HTML（如 p 元素）标记这些文本。

(3) 输入 </pre>。

提 示 预格式化的文本通常以等宽字体（如 Courier、Courier New 等）显示，参见图 4.14.2，可以使用 CSS 改变字体样式（参见第 10 章）。

① ASCII 艺术指的是用计算机字符（主要是 ASCII 字符）表示图片的一种艺术形式，通常要求使用等宽字体显示。——译者注

> 提示　如果要显示包含HTML元素的内容(如教程中的代码示例)，应将包围元素名称的 < 和 > 分别改为其对应的字符实体 < 和 >(有关示例参见 4.13 节)。否则，浏览器就会试着显示这些元素。一定要对页面进行验证，检查是否在 pre 中嵌套了 HTML 元素(参见 20.5 节)。

> 提示　不要将 pre 作为逃避以合适的语义标记内容和用 CSS 控制样式的快捷方式。例如，如果你想发布一篇在字处理软件中写好的文章，不要为了保留原来的格式，简单地将它复制、粘贴到 pre 里。相反，应该使用 p(以及其他相关的文本元素)标记内容，编写 CSS 控制页面的布局。

> 提示　同段落一样，pre 默认在新的一行显示，参见图 4.14.2。

pre 在表现方面的注意事项

注意，浏览器通常会对 pre 里面的内容关闭自动换行，因此，如果这些内容很宽，就会影响页面的布局，或产生横向滚动条。下面的 CSS 规则可以对 pre 的内容打开自动换行，但在 Internet Explorer 7 及以下版本中并不适用。

```
pre {
    white-space: pre-wrap;
}
```

与之相关的一点提示是，在大多数情况下不推荐对 div 等元素使用 white-space: pre 以代替 pre，因为空格可能对这些内容(尤其是代码)的语义非常重要，而只有 pre 才能始终保留这些空格。(同时，如果用户在其浏览器中关闭了 CSS，格式就丢失了。)

我们将从第 7 章开始讲解 CSS。文本的格式化将在第 10 章中讨论。

4.15　指定细则

根据 HTML5，small 表示细则一类的次要的注释，“通常包括免责声明、注意事项、法律限制、版权信息等。有时还用来署名，或用来满足许可要求。”small 通常是行内文本中的一小块，而不是包含多个段落或其他元素的大块文本(参见图 4.15.1 和图 4.15.2)。

```
...
<body>

<p>Order now to receive free shipping.
→<small>(Some restrictions may apply.)
→</small></p>

...

<footer>
    <p><small>&copy; 2011 The Super
    →Store. All Rights Reserved.
    →</small></p>
</footer>

</body>
</html>
```

图 4.15.1　在下面的两个例子中，small 元素都用以表示简短的法律声明。在第二个例子中，small 表示的是包含在页面级 footer 里的版权声明，这是一种常见的用法



图 4.15.2　在一些浏览器中，small 元素中文本的字号会比普通文本的字号小，不过，视觉上的大小与是否该用它标记内容是毫无关系的

指定细则的步骤

(1) 输入 <small>。

(2) 输入表示免责声明、注解、署名等类型的文本。

(3) 输入 </small>。

提示 一些浏览器会减小 small 文本的字号（参见图 4.15.2）。不过，一定要在 small 符合内容语义的情况下使用该元素，而不是为了减小字号而使用。总是可以用 CSS 控制字号（如果你愿意，甚至可以让它的字号变大）。更多信息参见 10.5 节。

提示 用 small 标记页面的版权信息是一种常见的做法（图 4.15.1 和图 4.15.2）。不过，small 只适用于短语，因此不要用它标记长的法律声明，如“使用条款”和“隐私政策”页面。这些内容应该用段落和其他需要的语义进行标记。

4.16 创建换行

浏览器会根据包含内容的块或窗口的宽度让文本自动换行。大多数情况下，让内容像这样充满整行是很合适的，但有时你希望手动地强制文字进行换行。可以使用 br 元素实现这一要求。

要确保使用 br 是最后的选择，因为该元素将表现样式带入了 HTML，而不是让所有的呈现样式都交由 CSS 控制。例如，不要使用 br 模拟段落之间的距离。相反，应该用 p 标记两个段落并通过 CSS 的 margin 属性规定两段之间的距离。

那么，什么时候该用 br 呢？实际上，对于诗歌、街道地址（参见图 4.16.1 和图 4.16.2）等应该紧挨着出现的短行，都适合用 br 元素。

```
...
<body>

<p>53 North Railway Street<br />
Okotoks, Alberta<br />
Canada T1Q 4H5</p>

<p>53 North Railway Street <br />Okotoks,
→ Alberta <br />Canada T1Q 4H5</p>

</body>
</html>
```

图 4.16.1 同样的地址出现了两次，不过出于演示的目的，对它们的编码有所不同。记住，代码里的回车总是会被忽略的，因此两个段落的显示是一样的，参见图 4.16.2。此外，在 HTML5 中书写 br 既可以使用
 也可以使用

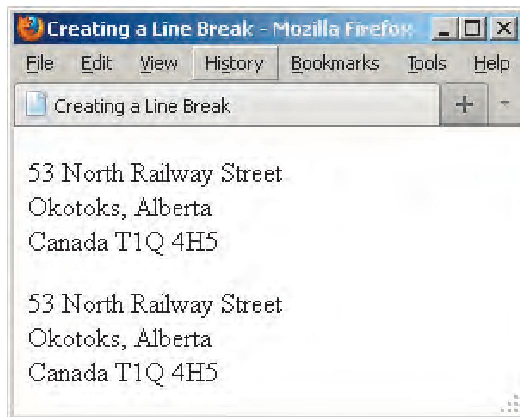


图 4.16.2 每个 br 元素强行让接下来的内容在新的一行显示

插入换行的方法

在需要换行的地方输入
（或
）。没有单独的 br 结束标记，因为它是所谓的空元素，没有任何内容。

提示 在 HTML5 中，输入
 或
 都是有效的。

提 示　可以使用样式表控制段落中的行间距(参见 10.6 节) 以及段落之间的距离（参见 11.8 节）。

提 示　hCard 微格式（http://microformats.org/wiki/hcard)是“用于表示人、公司、组织和地点”的人类和机器都可读的语义形式。可以使用微格式替代图 4.16.1 中表示街道地址的方式。

4.17　创建 span

同 div 一样，span 元素是没有任何语义的。不同的是，span 只适合包围字词短语，而 div 适合包含块级内容（参见 3.13 节）。

如果你想将下面列出的项目应用到某一小块内容，而 HTML 又没有提供合适的语义化元素，就可以使用 span。

- ❑ 属性，如 class、dir、id、lang、title 等（参见图 4.17.1 和图 4.17.2）。
- ❑ CSS 样式。
- ❑ JavaScript 行为。

由于 span 没有任何语义，因此应将它作为最后的选择，仅在没有其他合适的元素时才使用它。

```
...
<body>

<h1 lang="es">La Casa Milà</h1>

<p>Gaudí's work was essentially useful.
→ <span lang="es">La Casa Milà</span> is
→ an apartment building and <em>real people
→ </em> live there.</p>

</body>
</html>
```

图 4.17.1　在这个例子中，我想对一小块文字指定不同的语言，但从句子的上下文看，没有一个语义上适合 La Casa Milà 的 HTML 元素。段落之前的 h1 包含 La Casa Milà，它在语义上是合适的，因为这些文字就是后面内容的标题。因此对标题来说，直接在 h1 中添加 lang 属性就可以了，不必为了指定语言而包一个 span



图 4.17.2　span 元素没有任何默认样式

添加 span 的步骤

(1) 输入 <span。

(2) 如果愿意，输入 id="*name*"，其中 *name* 用于唯一地标识 span 包含的内容。

(3) 如果愿意，输入 class="*name*"，其中 *name* 是 span 所属类的名称。

(4) 如果愿意，输入其他的属性（如 dir、lang、title 等）及其值。

(5) 输入 > 结束开始标记。

(6) 创建希望包含在 span 里的内容。

(7) 输入 。

提 示　span 没有任何默认格式，参见图 4.17.2，但就像其他 HTML 元素一样，可以用 CSS 添加你自己的样式(参见第 10 章和第 11 章)。

提 示　可以对一个 span 元素同时添加 class 和 id 属性，但通常只应用这两个中的一个(如果真要添加的话）。主要区别在于，class 用于一组元素，而 id 用于标识页面中单独的、唯一的元素。

提 示　在 HTML 没有提供合适的语义化元素时，微格式经常使用 span 为内容添加语义化类名，以填补语义上的空白。要了解更多信息，参见 http://microformats.org。

4.18 其他元素

本节讲的是可以用到文本中的其他元素，但它们通常只在极少数情况下才会用到，或者浏览器对它们的支持还不完善（或兼有之）。

1. u 元素

同 b、i、s 和 small 一样，u 元素在 HTML5 中被重新定义了，使之不再是无语义的、用于表现的元素。以前，u 元素用于为文本添加下划线。HTML5 对它的定义为：

u 元素为一块文字添加明显的非文本注解，比如在中文中将文本标明为专有名词（即中文的专名号[①]），或者标明文本拼写有误。

下面是使用 u 标注拼错的词的例子：

```
<p>When they <u class="spelling">
→recieved</u> the package, they put
→it with <u class="spelling">there
→</u> other ones with the intention
→of opening them all later.</p>
```

class 完全是可选的，它的值（可以是你想填的任何内容）也不会在内容中明显指出这是个拼写错误。不过，可以用它对拼错的词添加不同于普通文本的样式（u 默认仍以下划线显示）。也可以通过 title 属性添加注释，如 [sic]（在一些语言中用于指示拼写错误的惯用符号）。

仅在 cite、em、mark 等其他元素语义上不合适的情况下使用 u 元素。同时，最好改变 u 文本的样式，以免与同样默认添加下划线的链接文本弄混，参见图 4.18.1。

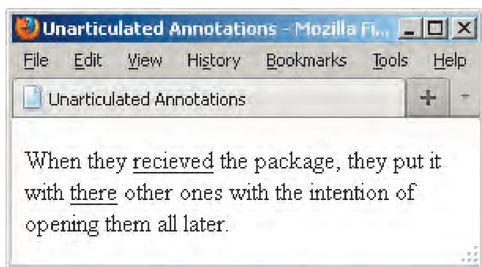


图 4.18.1 同链接一样，u 元素默认添加下划线。除非用 CSS 改变它们的样式，否则容易引起混淆

2. wbr 元素

HTML5 为 br 引入了一个近亲元素，称为 wbr。它代表"一个可换行处"。可以在一个较长的无间断短语（如 URL）中使用该元素，表示此处可以在必要的时候进行换行，从而让文本在有限的空间内更具可读性。因此，与 br 不同，wbr 不会强制换行，而是让浏览器知道哪里可以根据需要进行换行。

下面是一些例子：

```
<p>They liked to say,
"FriendlyFleasandFireFlies<wbr />
→FriendlyFleasandFireFlies<wbr />
→FriendlyFleasandFireFlies<wbr />
→"as fast as they could over and
→over.</p>
<p>His favorite site is this<wbr
/>is<wbr />a<wbr />really<wbr
/>really<wbr />longurl.com.</p>
```

输入 wbr 时，既可以用 <wbr />，也可以用 <wbr>。你或许已经猜到了，用到 wbr 的场合并不多。而且，截至本书写作之际，浏览器对它的支持并不一致。尽管在当前版本的 Chrome 和 Firefox 中 wbr 是有效的，但 Internet Explorer 和 Opera 会忽略它。

① 专名号用于表示人名、地名、朝代名等专名。——译者注

3. ruby、rp 和 rt 元素

旁注标记（ruby annotation）是东亚语言（如中文和日文）中一种惯用符号，通常用于表示生僻字的发音。这些小的注解字符出现在它们标注的字符的上方或右方。它们常简称为旁注（ruby 或 rubi）。日语中的旁注字符称为振假名（furigana）。

ruby 元素以及它们的子元素 rt 和 rp 是 HTML5 中为内容添加旁注标记的机制。rt 指明对基准字符进行注解的旁注字符。可选的 rp 元素用于在不支持 ruby 的浏览器中的旁注文本周围显示括号。

下面的例子用英文占位符展示了这种结构，帮助你理解代码中和支持它们的浏览器中对各类信息的安排方式，参见图 4.18.2。旁注文本已被突出显示：

```
<ruby>
    base <rp>(</rp><rt>ruby chars
    → </rt><rp>)</rp>
    base <rp>(</rp><rt>ruby chars
    → </rt><rp>)</rp>
</ruby>
```

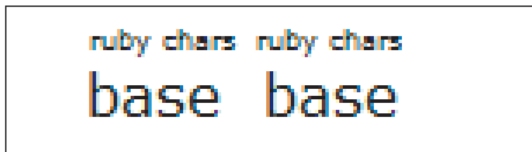


图 4.18.2　支持旁注标记的浏览器会将旁注文本显示在基准字符的上方（也可能在旁边），不显示括号

现在，展示一个真实例子。例子中有两个表示 Beijing 的中文基准字符，以及伴随它们的旁注字符（参见图 4.18.3）：

```
<ruby>
    北 <rp>(</rp><rt> ㄅㄟˇ</rt><rp>)
    → </rp>
    京 <rp>(</rp><rt> ㄐㄧㄥ </rt><rp>)
    → </rp>
</ruby>
```

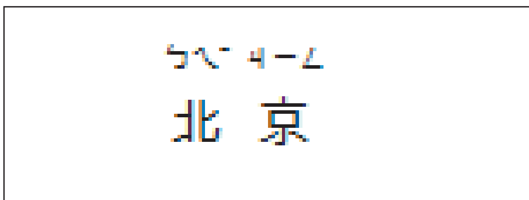


图 4.18.3　支持旁注的浏览器中显示的 Beijing 的旁注标记

可以看到在不支持 ruby 的浏览器中括号的重要性，参见图 4.18.4。没有它们，基准字符和旁注文本就会显示在一起，让内容变得混乱。

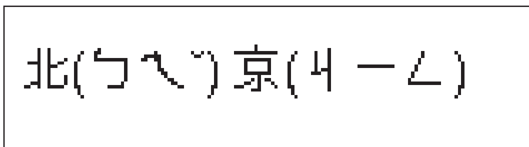


图 4.18.4　支持旁注的浏览器会忽略 rp 括号，仅显示 rt 内容（参见图 4.18.2 和图 4.18.3）。不过，不支持旁注的浏览器会在括号中显示 rt 内容，如这里所示

提示　在本书写作之际，只有 Safari 5+、Chrome 11+ 和 Internet Explorer 所有版本拥有基本的 ruby 支持（这也是使用 rp 的原因）。不过，Firefox 插件 HTML Ruby（https://addons.mozilla.org/en-US/firefox/addon/html-ruby/）为 Firefox 提供了对旁注的支持。

提示　关于旁注字符的更多信息，参见 http://en.wikipedia.org/wiki/Ruby_character。

4. bdi 和 bdo 元素

如果你的 HTML 页面中混合了从左至右书写的字符（如大多数语言所用的拉丁字符）和从右至左书写的字符（如阿拉伯语或希伯来语字符），就可能要用到 bdi 和 bdo 元素。

不过，首先要讲一点背景知识。除非在 html 元素中添加 dir 属性并将属性值设为 rtl，否则内容的基准方向都默认为从左至右。例如，<html **dir="rtl"** lang="he"> 指明内容的方向为从右至左，且基准语言为希伯来语。

就像我在全书多个示例中对 lang 的处理方式一样，对于页面内的元素，如果其内容与页面基准设置不一致，也可以对其设置 dir 属性。因此，如果基准语言设为英语（<html lang="en">），又要包含一段希伯来语，就可以标记为 <p dir="rtl" lang="he">...</p>。

在单独进行了设置的地方，内容一般将按照希望的方向显示。这是由 Unicode 的双向（bidi）算法控制的。

当上述算法不按设想的方式显示文本时，可以使用 bdo（bidirectional override，双向重写）元素对内容进行重写。通常，这代表 HTML 源代码中的内容是视觉顺序而非逻辑顺序的情况。

视觉顺序（visual order）就是内容看上去的顺序——HTML 源代码内容与你希望显示的顺序相同。*逻辑顺序*（logical order）则与之相反，可用于希伯来语这样从右至左书写的语言；先输入从右至左的第一个字符，然后是第二个（也就是第一个字符左边的字符），以此类推。

根据最佳实践，Unicode 希望双向文本都以逻辑顺序呈现。因此，如果文本是以视觉顺序呈现的，算法仍会让字符反向，显示出与预期相反的顺序。如果你无法将 HTML 源代码中的文本转换为逻辑顺序（例如，这些文本可能来自数据库或源），就只能依靠 bdo 了。

要使用 bdo，必须包含 dir 属性并将属性值设为 ltr（左至右）或 rtl（右至左），指定你希望呈现的方向。继续先前在英文页面中包含希伯来语段落的例子，应输入 <p lang="he">**<bdo dir="rtl">...</bdo>**</p>。bdo 适用于段落里的短语或句子。不能用它包围多个段落。

bdi 元素是 HTML5 中新加的元素，用于内容的方向未知的情况。不必包含 dir 属性，因为默认已设为自动判断。下面的例子来自 HTML5 规范（但有少许修改）：

这个元素特别适用于包围方向未知的用户生成的内容。

在这个例子中，用户名与用户提交的帖子的数量显示在一起。如果不使用 bdi 元素，阿拉伯用户的用户名将让文本变得难以理解（双向算法会把冒号和数字 3 置于 User 一词旁边，而不是在 posts 一词旁边）。

```
<ul>
    <li>User <bdi>jcranmer</bdi>:12 posts.</li>
    <li>User <bdi>hober</bdi>:5 posts.</li>
    <li>User <bdi>إيان</bdi>:3 posts.</li>
</ul>
```

提 示 *要了解关于右至左语言这一主题的更多信息，推荐阅读 W3C 的文章“Creating HTML Pages in Arabic, Hebrew, and Other Right-to-Left Scripts”（www.w3.org/International/tutorials/bidi-xhtml/）。*

---

5. meter 元素

meter 元素也是 HTML5 的新元素。可以用它表示分数的值或已知范围的测量结果。简单地说，它代表的是投票结果（例如，“30% Smith，37% Garcia，33% Clark”）、已售票数（例如，“共 850 张，已售 811 张”）、考试分数、磁盘使用量等测量数据。

HTML5 建议浏览器在呈现 meter 时，在旁边显示一个类似温度计的图形——一个表示测量值的横条，测量值的颜色与最大值的

有所区别（当然，除非它们相等）。作为当前少数几个支持 meter 的浏览器，Chrome 正是这样显示的，参见图 4.18.5。对于不支持 meter 的浏览器，可以通过 CSS 对 meter 添加一些额外的样式，或用 JavaScript 进行改进。

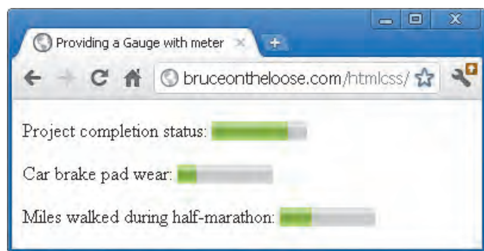


图 4.18.5　Chrome 等支持 meter 的浏览器会自动显示测量值，并根据属性值进行着色。<meter> 和 </meter> 之间的文字不会显示出来

尽管不是必需，不过最好在 meter 里包含一些反映当前测量值的文本，供不支持 meter 的浏览器显示，参见图 4.18.6。

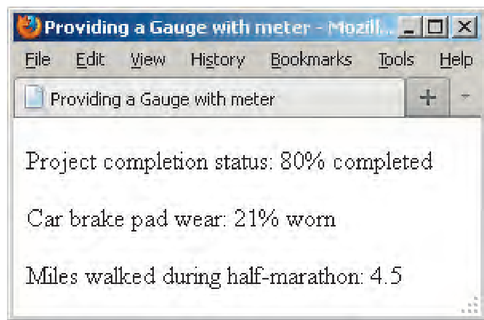


图 4.18.6　大多数浏览器（如 Firefox）不支持 meter，它们会将 meter 元素里的文本内容显示出来。可以通过 CSS 改变其外观

下面是一些 meter 的例子（如图 4.18.5 和图 4.18.6 所示）：

```
<p>Project completion status: <meter
➝value="0.80">80% completed</meter>
➝</p>
<p>Car brake pad wear: <meter low=
➝"0.25" high="0.75" optimum="0"
➝value="0.21">21% worn</meter></p>
<p>Miles walked during half-marathon:
➝<meter min="0" max="13.1" value="4.5"
➝title="Miles">4.5</meter></p>
```

meter 不提供定义好的单位，但可以使用 title 属性指定单位，如最后一个例子所示。Chrome 会以工具提示的方式显示它，参见图 4.18.5。

提示　meter 支持好几个属性。value 是其中唯一必需包含的属性。如果不指定 min（最小值）和 max（最大值），则默认将它们分别设为 0 和 1.0。low、high 和 optimun 属性通常共同作用，它们将范围划分为低、中、高三个区间。optimum 代表范围内的最优位置，如上文例子中的 0 brake pad wear。如果 low 和 high 的值均不是最优的，可以将 optimum 设为它们之间的值。

提示　截至本书写作之际，仅有 Chrome 11+ 和 Opera 11+ 支持 meter。这大概也是现实中很少见到它的原因。随意使用它，只是需要了解大多数浏览器默认只显示 meter 文本而不显示图形。

提示　每个支持 meter 的浏览器显示测量值图形的样式可能有差异。

提示　已经有人试过针对支持 meter 的浏览器和不支持的浏览器统一编写 meter 的 CSS。在网上搜索 style HTML5 meter with CSS（用 CSS 为 HTML5 的 meter 添加样式），就可以找到一些解决方案（注意其中的一些用到了 JavaScript）。

提示　meter 并不用于标记没有范围的普通测量值，如高度、宽度、距离、周长等。例如，这种写法是不正确的：<p>I walked <meter value="4.5">4.5</meter> miles yesterday.</p>。

提示　一定不要将 meter 和 progress 元素混在一起使用。

6. progress 元素

progress 元素也是 HTML5 的新元素。可以用它表示一个进度条，就像在 Web 应用中看到的指示保存或加载大量数据操作进度的那种组件。

就像 meter 一样，支持 progress 的浏览器会根据属性值自动显示一个进度条，参见图 4.18.7。同时，和 meter 一样，为了让旧的浏览器也能表现进度，最好在 progress 中包含反映当前进度的文本（参见图 4.18.8），尽管这并不是必需的。

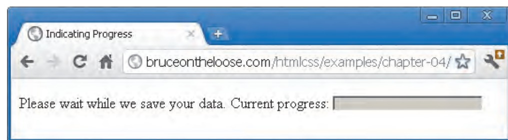


图 4.18.7　Chrome 等支持 progress 的浏览器会自动显示进度条，并根据值对其进行着色。<progress> 和 </progress> 之间的文本不会显示出来。在这个例子中 value 属性设成了 0，因此整个横条都是同样的颜色

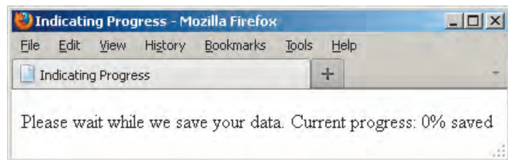


图 4.18.8　Firefox 不支持 progress，因此不会显示有颜色的横条，而是显示该元素里面的文本。可以通过 CSS 改变其外观

下面是一个例子：

```
<p>Please wait while we save your
→data. Current progress: <progress
→max="100" value="0">0% saved
→</progress></p>
```

对 progress 的完整说明已经超出了本书的范围。通常，你只能通过 JavaScript 动态地更新 value 属性值和元素里面的文本以指示任务进程（例如，指示已完成 37%）。使用 JavaScript 与在 HTML 中硬编码（即 <progress max="100" **value="37">37% saved** </progress>）从视觉上看是一样的，参见图 4.18.9。当然，不支持的浏览器会显示为图 4.18.8 所示的样子。

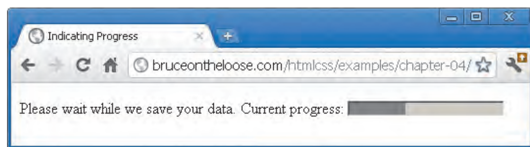


图 4.18.9　Chrome 中显示的进度条，通过 JavaScript（或直接在 HTML 中）将 value 属性设为 37（假定 max="100"）

提示　progress 元素支持三个属性：max、value 和 form。它们都是可选的，max 属性指定任务的总工作量，其值必须大于 0。value 是任务已完成的量。如果 progress 没有嵌套在一个 form 元素里面，又需要将它们联系起来，可以添加 form 属性并将其值设为该 form 的 id。

提示　下面简单地看看如何用 JavaScript 修改 progress 元素。假定进度条已经设定了一个 id，如：

```
<progress max="100" value="0"
→ id="progressBar">0% saved</progress>
```

下面的 JavaScript 可以让你获取该元素：

```
var bar = document.getElementById
→ ('progressBar');
```

然后就可以通过 bar.value 取出或设定 value 的值。例如，bar.value=37; 可以将值设为 37。

提示　在本书写作之际，progress 元素已在现代浏览器中获得了相当坚实的支持。支持它的浏览器包括 Chrome 11+、Firefox 6+、Internet Explorer 10（在本书写作时仅有平台预览版）和 Opera 11+，Safari 还不支持该元素。

提示　每个浏览器显示进度条的样式可能不同，不过你可以通过 CSS 为它应用样式。

第 5 章

图　　像

5

本章内容

- ❑ 关于 Web 图像
- ❑ 获取图像
- ❑ 选择图像编辑器
- ❑ 保存图像
- ❑ 在页面中插入图像
- ❑ 提供替代文本
- ❑ 指定图像尺寸
- ❑ 在浏览器中改变图像的尺寸
- ❑ 在图像编辑器中改变图像的尺寸
- ❑ 为网站添加图标

为 Web 创建图像与为在纸上输出而创建图像有所不同。尽管 Web 图像和可打印图像的基本性质是相同的，但它们在格式、颜色、尺寸 / 分辨率、速度、透明度和动画等六个主要方面有一些区别。本章将阐释这六个方面的重要要素，以及如何使用这些知识为网站创建有效的图像。

创建图像以后，就要将它们插入到网站中去。

5.1　关于 Web 图像

让我们来看看创建 Web 图像时应记住的六个要素。

1. 格式

在纸上打印图像的人不必担心读者将使用什么查看图像。但 Web 设计人员就要考虑这个问题。每天，人们通过数以百万计的 Mac、基于 Windows 的 PC、手机、平板电脑等各种各样的设备访问 Web。网站中的图像应该采用这些操作系统都能识别的格式。当前，Web 上用的最广泛的三种格式是 GIF、PNG 和 JPEG。当前的浏览器都能查看这三种图像格式。

你应该选择质量最高，同时文件最小的格式。

JPEG 格式适用于彩色照片，因为它包含大量的颜色并进行了合理的压缩，使文件变得比较小，参见图 5.1.1。它是一种有损的格式，因此在将图像保存为 JPEG 时会丢失一部分原始信息，但通常这样做是值得的，因为你的页面可以更快地加载。我们将在“速度”一节继续讨论这个问题。



图 5.1.1　全彩的照片应保存为 JPEG 或 PNG-24 格式（另见彩插）

PNG 和 GIF 格式通常用于保存拥有大量纯色和图案或有透明度的标志之类的文件。对于连续的颜色或重复的图案，PNG 和 GIF 格式的压缩效果要好于 JPEG 的。选择 PNG 通常更好一些，因为它对小文件的压缩算法更好，且具有更高级的透明度支持（alpha 透明度），参见图 5.1.2。



图 5.1.2　标志和计算机生成的其他图像（即只有少数几种颜色的图像）可通过 ZIP 有效地压缩，因此通常保存为 PNG-8 格式（另见彩插）

2. 颜色

大多数计算机显示器可以显示数以百万计的颜色，但也有例外的情况。有的图像格式的调色板是有限的。GIF 和 PNG-8 图像只能拥有 256 种颜色（对标志和图标来说这已经够了）。

照片和复杂的插图应保存为 JPEG 或 PNG-24 格式，因为它们要在单张图片中包含更多的颜色。

3. 尺寸和分辨率

数字图像以像素为单位进行度量。300 万像素的数码相机可以照出 2048 像素宽、1536 像素高的照片。这有多大呢？视情况而定。如果用 256 ppi（pixels per inch，像素每英寸）的打印机打印图像，它的尺寸就是 8 英寸 ×6 英寸（约 20 厘米 ×15 厘米）。但如果在网页上使用这个图像，它的尺寸就取决于访问者显示器的分辨率了。分辨率大约为 86 ppi（可能低至约 72 ppi，或高至约 100 ppi），这时图像将显示为 28 英寸 ×21 英寸（约 71 厘米 ×53 厘米）。这太大了，参见图 5.1.3。

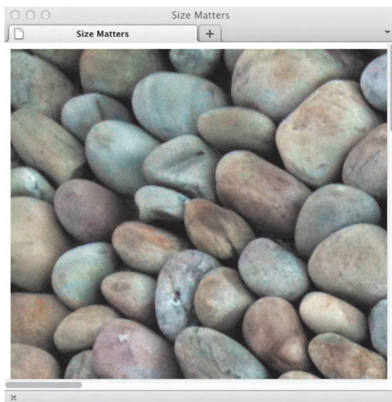


图 5.1.3　这个图像为 2048 像素宽。在 Photoshop 中，使用 256 ppi 的输出分辨率，它只有 6 英寸 ×8 英寸（约 15 厘米 ×22 厘米）。而在这里的 Firefox 中，分辨率是由访问者的显示器决定的。假定分辨率为 72 ppi，该图就有 28 英寸（约 71 厘米）宽

也许，最好根据平均网页宽度考虑图像的尺寸。由于长期以来 1024 像素 ×768 像素的屏幕分辨率都是标准的分辨率，网页设计人员已习惯于让页面保持 960 像素宽，从而避免产生横向滚动条，让访问者可以看到整个页面的内容（参见图 5.1.4）。

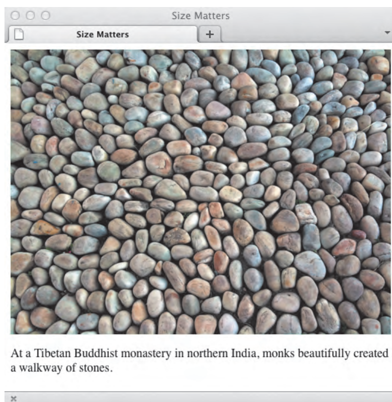


图 5.1.4　这个图像有 500 像素宽，大约为 1024 像素宽的浏览器窗口宽度的一半

事实上，已有越来越多的人开始使用更大的显示器（现在，超过85%的分辨率大于1024像素 ×768像素），但这并不意味着人们会让单个浏览器窗口填满这些更大的显示器。如果不查看其他的程序（或其他的浏览器窗口），在过宽的浏览器中阅读文本是一件很累的事。不过，仍有设计人员倾向于扩宽他们的设计，并使用宽度灵活的设计，让内容可随着浏览器窗口放大或缩小。

同时，智能手机和平板电脑的使用已变得越来越多，因此应该始终考虑屏幕大小和受限的下载速度。

这里，分辨率可以代表两种不同的概念：一个是显示器或图像中的实际像素数量（如640×480），一个是显示器或图像上1英寸的像素数量（如72或86 ppi）。不管怎样，分辨率越高，像素越多。在纸上，像素可以增加细节或尺寸。在屏幕上，则总是像素越多，图像就越大。

4. 速度

Web图像与可打印图像的另一项区别在于Web访问者必须等待图像下载（试想等待早报上的图像逐渐显现的情形！）。

如何让下载时间最短呢？最容易的办法就是使用小图像。图像文件越大，访问者看到它之前所花的时间就越长。

第二种提高下载速率的方式就是对图像进行压缩。JPEG的强大之处在于它可以大幅降低文件的大小，但JPEG有两个主要的缺点。首先，它的压缩信息占用大量空间，因此不适合小图像。其次，它是有损压缩——为了节省空间，可能会永久性地牺牲一些细节。对图像进行解压也找不回丢失的数据。如果想在未来某个时候对图像进行编辑，就需要用无压缩格式（如PSD或TIFF）进行保存，直到编辑完成以后再保存为JPEG。

PNG和GIF是无损的格式，因此它们可以在保证质量的情况下压缩图像。拥有大片单色区域的图像（如标志、显示的文本、插图等）最适合用这两种格式。此外，PNG的压缩质量比GIF好一些。

5. 透明度

出于两个原因，透明度很重要。首先，使用透明度将一个图像置于另一个图像的后面可以创建复杂的布局。其次，可以利用透明度为图像创建非矩形的边缘，增强页面的视觉吸引力。PNG和GIF都支持透明度，但JPEG不支持。

在GIF格式中，一个像素要么是透明的，要么是不透明的。而PNG则支持alpha透明。alpha透明既支持全透明，又支持半透明。这意味着具有复杂透明背景的图像使用PNG的效果要好于使用GIF的效果，因为使用PNG可以让边缘变得平滑，避免产生锯齿。

6. 动画

纸上永远也看不到的一样东西就是动画。但在万维网上，它们随处可见。动画可以保存为GIF，但不能是JPEG或PNG。

在图像内显示动画已经用得越来越少了。通常可以使用Flash、CSS动画和JavaScript创建动画。近几年，由于iOS不支持Flash，且JavaScript和CSS的功能逐渐变强，因此万维网上用Flash创建的动画已经变少了。

5.2　获取图像

如何获取可用在网页中的图像呢？有几种方式。可以购买或下载现成的图像，通过扫描仪将照片或手绘图像数字化，使用数码相机，或使用Adobe Photoshop这样的图像编辑软件从头绘制图像。在有了图像以后，就可以根据需要在万维网上使用它们了。

获取图像的手段

- ❑ 可以使用Google寻找万维网上的图像，方法是点击搜索框上方的“图像”（Images）链接，再输入搜索条件，跟平常一样。关于图像版权的更多信息，参见后面的“知识共享许可协议”。
- ❑ 一般来说，即使是在网上找到的免费图像，也会受到某种限制（参见“知识共享许可协议”）。不过，购买的图像通常能够以任何方式使用（除了对图像本身再次进行销售）。请仔细阅读声明或许可协议。
- ❑ 很多公司以较低的价格销售库存的照片和图像。通常，每张图像都有几个版本，以满足不同的目的和分辨率要求。
- ❑ 扫描仪和数码相机是创建你自己的图像的常用方法。

知识共享许可协议

知识共享（Creative Commons，www.creativecommons.org）是一个非营利组织，它开发了一个版权模板体系，让艺术家可以按照其指定的方式分享他们的作品，同时无需放弃对作品的所有权利。网站设计师、音乐家和摄影师使用知识共享许可协议将他们的作品放入市场，而无需担心别人以他们不允许的方式利用这些作品。

Flickr是流行的图片分享Web应用（www.flickr.com）。它要求用户为他们上传的每张图片指定一种知识共享许可协议。Flickr让访问者可以根据许可协议的类型搜索图片。它是为网站寻找图片的好地方。

使用Google时，可以根据使用权限对搜索进行限定（点击“高级搜索”[①]，然后在“使用权限”下拉菜单中选择需要的选项）。

5.3　选择图像编辑器

有很多不同的软件可以用来创建和保存Web图像。大多数现代图像编辑器都提供了创建Web图像的专用工具。这些工作考虑了本章前面讨论的几种要素。

毫无疑问，行业标准是Photoshop和它的“近亲”Adobe Fireworks（www.adobe.com）。Fireworks本身也非常强大。它们都既有Macintosh版也有Windows版。我用这两个软件讲解本章的一些技术。

不过，需要强调一点，无论使用哪种软件，优化Web图像的基本策略始终不会改变。它们所用的命令名称可能有所不同，步骤也有多有少，但思路是一样的。

有很多程序可以代替Photoshop，如Paint.NET（适合Windows，www.getpaint.net）以及Acorn或Pixelmator（适合Mac，www.pixelmator.com）。同时，Photoshop.com、Aviary.com等在线编辑器也变得越来越强大。尽量使用你感觉最舒服的程序。

5.4　保存图像

创建图像以后，要保存它们。这一过程是在图像的视觉质量与文件大小之间寻找平衡的艺术。

如果你的计算机上没有安装Photoshop或Fireworks，可以先使用它们的试用版。

1. Adobe Photoshop

Photoshop在File菜单中提供了Save for Web & Devices（存储为Web和设备所用格式）命令。它让用户可以从视觉上对比原始图像和一至三个优化后的版本，还可以看到对文件大小和下载时间的影响。

① 在新的Google首页界面中，要先点击右上角的齿轮按钮，才能看见“高级选项”。——译者注

使用 Photoshop 的 Save for Web & Devices 的步骤

(1) 打开 Photoshop，创建图像。或者打开现有的图像，并通过剪裁、调整大小和编辑，为发布做好准备。

(2) 选择 File → Save for Web & Devices（文件 → 根据 Web 和设备保存）。出现 Save For Web & Devices 对话框。

(3) 点击 2-Up（2 联）选项卡查看一个优化后的版本，或点击 4-Up（4 联）选项卡查看三个优化后的版本。

(4) 根据需要，点击一个优化后的版本。

(5) 选择希望采用的格式。

通常，在计算机上创建的图像（包括标志、横幅、线条艺术、文本以及任何包含由单色和锐利细节构成的大块区域的图像）应保存为 PNG-8 或 GIF 格式，参见图 5.4.1。

具有连续色调的图像（如照片）应保存为 JPEG 或 PNG-24 格式，参见图 5.4.2。

(6) 调整其他设置选项，直到在质量可接受的条件下获得最小的文件。

(7) 点击 Save。选择一个目录，并对新文件进行命名。它将自动包含所选格式的扩展名（而且，这样通常就不会覆盖原始图像）。

2. Adobe Fireworks

Photoshop 是为多种用途设计的，而 Fireworks 则是专门为创建 Web 图像设计的（参见图 5.4.3 和图 5.4.4）。过去，它对 PNG 的优化能力要强于 Photoshop，它优化后的文件更小，但目前这种差距已经被填平了。不过，Fireworks 提供了另一种输出高质量 PNG 的选项：PNG-32 格式。

提 示　要记住，你的主要目标是在维持可接受的图像质量的条件下，获得尽可能小的文件。

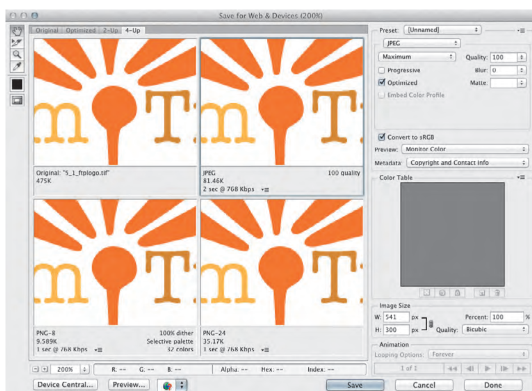


图 5.4.1　Save for Web & Devices 对话框。它显示原始图像（左上）和三个可能的压缩后的版本。这个图像有很多纯色，还有应该保持锐利的文本。可以看到，PNG-8 格式（左下）的压缩程度最高，压缩后不到 10K①。PNG-24 可用的颜色更多，压缩后为 35K。高质量的 JPEG 则非常大。如果将 JPEG 调整为中等质量（未显示），也仍然比较大，还比较难看（另见彩插）

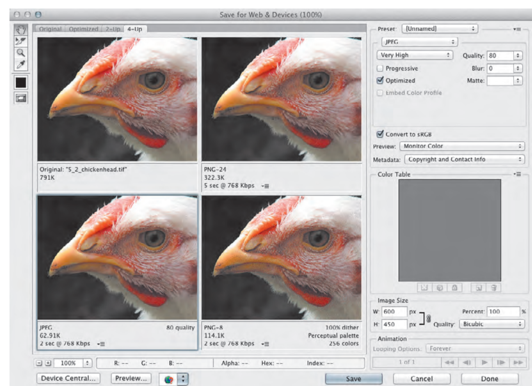


图 5.4.2　JPEG（左下）的图像质量最好，文件也最小（63K）。PNG-8 压缩只留下了照片的色带（右下），且文件大小（114K）也几乎是 JPEG 的两倍。PNG-24（右上）可以呈现高质量的图像，但文件也大得多（322K）（另见彩插）

① K 代表 KB（千字节）。——译者注



图 5.4.3　注意，PNG-8 图像（左下）比其 Photoshop 副本文件略小一些。而另两种格式，即 JPEG（右上）和 PNG-24（右下），则大一些（另见彩插）

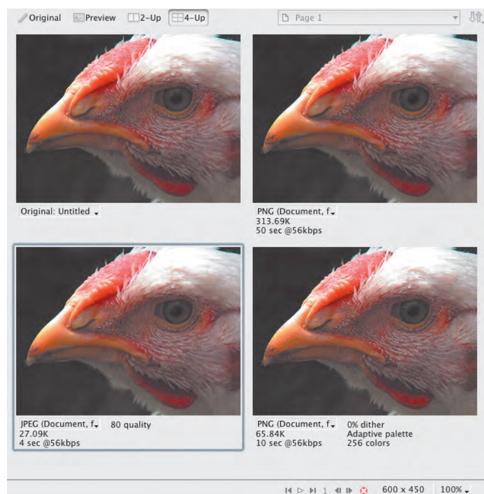


图 5.4.4　Fireworks 对图像的优化效果明显好一些，JPEG（左下）、PNG-8（右下）和 PNG-24（右上）的文件大小都更小一些。JPEG 和 PNG-8 文件大小只有 Photoshop 的一半（另见彩插）

提示　应该使用 RGB[1]模式创建图像，而不是 CMYK[2]模式（用于印刷）。

提示　如果不确定应该选择哪种格式，可以比较两种优化结果，看看哪种格式的压缩效果更好。

提示　PNG-24 是一种强大的无损格式，既适用于计算机生成的图像，又适用于“自然的”彩色图像。它通常比 PNG-8 好，但不如 JPEG。

提示　如果一个图像拥有两种类型的内容，可以将它分成两块，分别进行压缩，然后用 CSS 重新组装。当然，也可以只用一种格式，然后取其最优的格式。

提示　Save for Web & Devices 命令创建新的图像并保持原始图像不变，除非以相同的名称和扩展名保存于同一个文件夹下，这样就可以覆盖旧文件。

提示　在优化后的版本中，只保留图像可见的层[3]。

5.5　在页面中插入图像

可以在网页中放置各种各样的图像，从标志到照片都可以。只要浏览器没有关闭图像显示，当访问者进入你的页面时，页面上的图像就会自动显示出来（参见图 5.5.1 和图 5.5.2）。

① R、G、B 分别表示 Red（红）、Green（绿）、Blue（蓝）。RGB 模式又称三原色光模式。——译者注

② C、M、Y、K 分别表示 Cyan（青）、Magenta（品红）、Yellow（黄）、Black（黑）。CMYK 模式又称印刷四分色模式。——译者注

③ 编辑图像时，常常将图像的不同成分分层排列。优化后的版本会去除层的信息。——译者注

```
...
<body>

<h2>Barcelona's Market</h2>

<img src="cornermarket.jpg" />

<p>This first picture shows one of the
➝ fruit stands in the Mercat de la Boquería,
➝ the central market that is just off the
➝ Rambles. It's an incredible place, full
➝ of every kind of fruit, meat, fish, or
➝ whatever you might happen to need. It
➝ took me a long time to get up the nerve
➝ to actually take a picture there. You
➝ might say I'm kind of a chicken, but
➝ since I lived there, it was just sort
➝ of strange. Do you take pictures of your
➝ supermarket?</p>

</body>
</html>
```

图 5.5.1　这个图像的 URL 只包含文件名，没有路径，因此代表该图像位于与此网页相同的文件夹



图 5.5.2　图像贴向页面的左侧，与文本的对齐方式一致。使用 float（参见 11.10 节）等 CSS 属性可以改变对齐方式或让文字环绕图像

在页面中插入图像的步骤

(1) 在 HTML 代码中，将光标放在希望图片出现的位置。

(2) 输入 `<img src="image.url"`，其中 *image.url* 指示图像文件在服务器上的位置。

(3) 输入一个空格和 `/>`。

提示　图像必须先上传到服务器上，访问者才有可能看到它们。

提示　不要期望你的访问者会长时间等待页面加载和显示。可以对页面进行测试（别忘了你的连接速度可能比访问者的快）。如果你都等不下去，那么访问者也一样等不下去。另一种办法是为大图创建缩略图，让访问者可以通过链接选择查看大图。

提示　有一个已经废弃的 border 属性（border="*n*"，其中 *n* 是以像素为单位的宽度），它可以在图像周围添加或去除边框（特别是去除链接图像周围自动出现的边框）。更好的方法是使用样式表控制图像的边框以及所有其他的方面。

5.6　提供替代文本

如果访问者使用较大的屏幕和较快的网络，图像可能效果很好，但对于手持设备、手机用户，连接速度慢的用户以及盲人，效果就不那么好了，甚至可能导致明显的问题。可以为图像添加一段描述性文本，当图像出于某种原因不显示的时候，就将这段文字显示出来。同时，屏幕阅读器可以朗读这些文本。

提供图像无法显示时的替代文本的步骤

(1) 在 img 标记内，在 src 属性及其值的后面，输入 alt="。

(2) 输入图像出于某种原因没有显示时应该出现的文本（参见图 5.6.1 和图 5.6.2）。

(3) 输入 "。

```
...
<body>

<h2>Barcelona's Market</h2>

<img src="cornermarket.jpg" alt="Fruit Stand
→ in Market" />

<p>This first picture shows one of the
→ fruit stands in the Mercat de la Boquería,
→ the central market that is just off the
→ Rambles. It's an incredible place, full
→ of every kind of fruit, meat, fish, or
→ whatever you might happen to need. It
→ took me a long time to get up the nerve
→ to actually take a picture there. You
→ might say I'm kind of a chicken, but
→ since I lived there, it was just sort
→ of strange. Do you take pictures of your
→ supermarket?</p>

</body>
</html>
```

图 5.6.1　尽管从理论上说，替代文本的长度没有限制，但大多数浏览器不会为其进行自动换行。因此，应尽量让替代文本少于 50 个字符



图 5.6.2　在 Internet Explorer 中，替代文本出现在一个带红叉的小方块旁边。在其他浏览器中，替代文本是单独出现的

提 示　在 HTML5 中，img 元素必须包含 alt 属性。

提 示　JAWS 等屏幕阅读器可以将替代文本朗读出来，这样盲人或视障人士就可以了解图像的大致内容了。

提 示　如果图像与非可视用户无关，W3C 建议设置 alt=""。如果图像有标题，或周围的文本可以准确地描述图像，也可以让 alt 留空。

5.7　指定图像尺寸

有时，加载网页会先看到文本，等一小段时间以后图片开始加载时，文本跳到图片周围，留出可容纳图片的空间。出现这种现象，是因为在 HTML 中没有指定图像的尺寸。

当浏览器读到插入图像的 HTML 代码时，必须加载图片，从而了解它有多大，需要为它留出多大的空间。如果指定图像的尺寸，浏览器就可以预留空间，在图像加载的同时让文本显示在周围，保持布局的稳定。可以通过浏览器或图像编辑软件获取图像的精确尺寸。

浏览器还会根据在 HTML（或 CSS）中指定的尺寸，对图像进行放大或缩小。如果你想在不同的环境使用同一张图片，就可以利用这一特性。不过，如果对图像文件进行了编辑，改变了它的尺寸，要注意更新代码。

1. 在浏览器中查看图像尺寸

(1) 在图像上右击，出现背景弹出菜单，参见图 5.7.1。

(2) 选择 Properties（属性）或 View Image Info（查看图像信息）（具体选项取决于所使用的浏览器），参见图 5.7.1，出现的框中会以像素为单位显示图像的尺寸，参见图 5.7.2。



图 5.7.1　在浏览器中的图像上右击，出现背景弹出菜单。浏览器会提供审查图像，显示其属性，或获取其尺寸的方式

Type:	JPEG Image
Size:	Unknown (not cached)
Dimensions:	300px × 399px
Associated Text:	Fruit Stand in Market

图 5.7.2　这个框（它的外观取决于所用的浏览器）以像素为单位显示了图像的尺寸

2. 在 Photoshop 中查看图像尺寸

(1) 在 Photoshop 中打开图像。

(2) 让文档窗口具有足够的宽度，直到能在左下边沿看见文档信息栏。

(3) 点击文档信息栏，会出现一个显示图像信息的小方框，其中包括图像的尺寸，参见图 5.7.3。



图 5.7.3　在 Photoshop 中，点击文档窗口底部的文档信息栏，就可以查看图像的属性（如果没有显示文档信息栏，就把窗口拉宽一些）

3. 在 HTML 中指定图像尺寸

(1) 使用“在浏览器中查看图像尺寸”或“在 Photoshop 中查看图像尺寸”中介绍的方法，确定图像的尺寸。

(2) 在 img 标记中，src 属性的后面，输入 `width="x" height="y"`，采用第 (1) 步中确定的值，以像素为单位指定 *x* 和 *y*（分别代表图像的宽度和高度）的值。

```
...
<body>

<h2>Barcelona's Market</h2>

<img src="cornermarket2.jpg" alt="Fruit Stamd
→ in Market" width="300" height="399" />

<p>This first picture shows one of the
→ fruit stands in the Mercat de la Boquería,
→ the central market that is just off the
→ Rambles. It's an incredible place, full
→ of every kind of fruit, meat, fish, or
→ whatever you might happen to need. It
→ took me a long time to get up the nerve
→ to actually take a picture there. You
→ might say I'm kind of a chicken, but
→ since I lived there, it was just sort
→ of strange. Do you take pictures of your
→ supermarket?</p>

</body>
</html>
```

图 5.7.4　如果明确指定以像素为单位的高度和宽度，浏览器就不必花时间判断图像的尺寸，从而更快地将图像显示出来

提 示　width 和 height 属性不一定要反映图像的实际尺寸。

提 示　如果有几个尺寸相同的图像，可以通过样式表同时设置它们的高度和宽度。

提 示　在浏览器中，如果在单独的窗口中打开图像，就可以在标题栏中看到图像的尺寸，参见图 5.7.4 和图 5.7.5。

> **提 示**　在 Photoshop 或 Fireworks 中，可以选中整个图像，再在 Info（信息）面板中查看图像的尺寸。

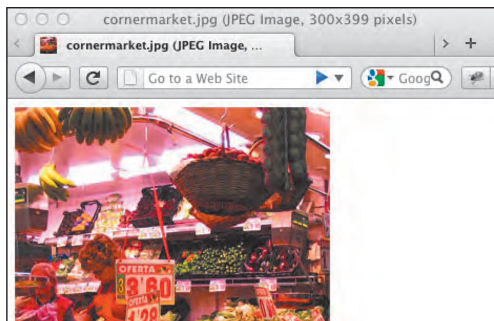


图 5.7.5　如果直接在浏览器（这里是 Windows 上的 Firefox）中打开图像，就会在标题栏里显示图像的尺寸

5.8　在浏览器中改变图像的尺寸

通过为图像指定新的高度和宽度（以像素为单位），可以改变图像显示的尺寸，参见图 5.8.1、图 5.8.2 和图 5.8.3。不过，最好还是以原始尺寸显示图像。



图 5.8.1　这个图像的原始尺寸为 440 像素 ×341 像素，这在页面上显得太大了

```
...
<h1>Stupas</h1>
<img src="stupa.jpg" alt="Two Stupas"
➝ width="220" height="170" />
<p>These stupas in Yunnan, China, are
➝ Buddhist monuments used as a place for
➝ worship.</p>
...
```

图 5.8.2　调整 height 和 width 属性，并确保比例不变。在这个例子中，height 和 width 都只有原来的一半

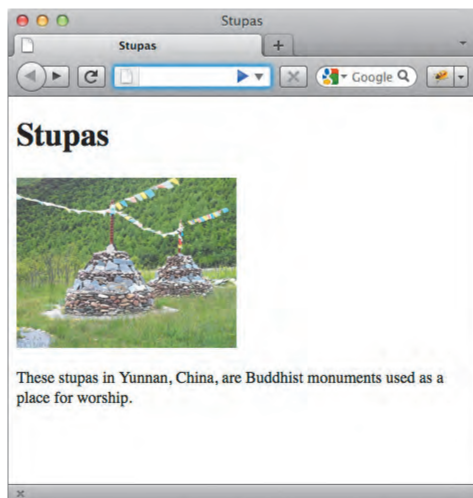


图 5.8.3　显示的图像只有原图的一半大小。不过要注意，加载它需要花的时间跟以前是一样的。毕竟，用的是同一个文件

浏览器使用的改变图像尺寸的方法并不比 Photoshop 或其他图像编辑器的方法先进，因此要对结果进行测试。

为减少加载时间，可以通过在 HTML 或 CSS 中调整高度和宽度改变图像的大小。不过要注意，如果将图像调得过大，就会显得粗糙难看。

在浏览器中改变图像尺寸的步骤

(1) 输入 `<img src="`*`image.url`*`"`，其中 *`image.url`* 是图像在服务器上的位置。

(2) 输入 width="*x*" height="*y*"，其中 *x* 和 *y* 分别是希望设定的宽度和高度（以像素为单位）。

(3) 根据需要，添加任何其他的图像属性，最后输入 />。

提示 在第 (2) 步中，也可以使用百分数。该数的基准是浏览器窗口的大小（而不是原始图像尺寸）。

提示 使用 width 和 height 属性是改变图像在网页上显示尺寸的一种快捷但不优雅的方式。由于文件本身并未改变，访问者实际上是被欺骗了。显示缩小的图像与原图相比要花去更多的下载时间；放大显示的图像则会变得粗糙。更好的方案是使用图像编辑器修改图像的尺寸。

提示 可以只设置 width 或只设置 height，让浏览器按比例自动调整另一个值。

5.9 在图像编辑器中改变图像的尺寸

大多数图像对网页来说都太大了。一幅用于打印的图像可能有 1800 像素宽（以 300 dpi[①]打印，有 6 英寸宽，约 15 厘米），而用于网页的图像很少需要超过 600 像素，通常仅为 200 像素左右。

当你需要将图像变大时，应寻找更大的原始图像，再用图像编辑器将它缩小到合适的尺寸。即使没法做到这样，使用图像编辑器将图像变大也比使用浏览器的效果更好。不过，这样会增加图像文件的大小，让页面加载时间变长。

用 Photoshop 改变图像尺寸

(1) 在 Save for Web & Devices 窗口右下角的 Image Size 部分，点击 W（宽度）框或 H（高度）框，如图 5.9.1 所示。

(2) 以像素为单位输入宽度或高度，或输入百分数，再按一下 Tab 键，改变图像的尺寸，如图 5.9.2 所示。

(3) 可以继续调大或调小，直到满意为止。在按下 Save 之前，图像不会重新取样。

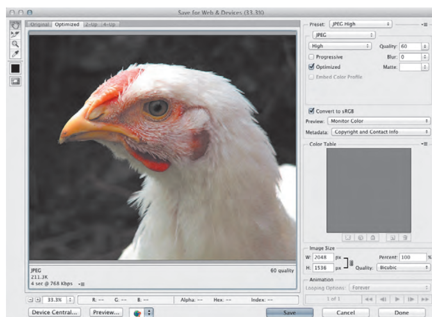


图 5.9.1 原始图像是用笔者的数码相机在默认设置下拍摄的，其尺寸为 2048 像素 ×1536 像素，即便对于四个浏览器窗口，也仍然很大。在压缩为高质量 JPEG 时，图像文件有 211.3K

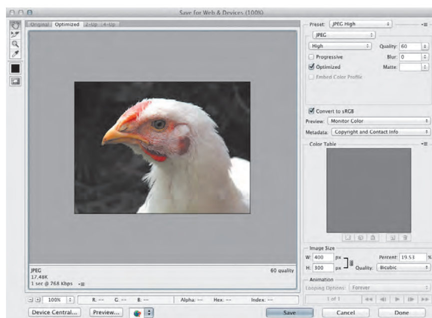


图 5.9.2 在 W 框中输入新的宽度 400 像素，再点击 Apply（应用）。缩小后的图像在页面中很合适，以 768 KB/S 的速率下载，只需要 1 秒（如果用更快的连接，时间会更短）

① dpi 即 dots per inch（点每英寸），指打印机在每英寸打印的点数。——译者注

提示　在进入 Save for Web & Devices 之前，也可以通过 Image（图像）菜单下的 Image Size（图像大小）命令改变图像的尺寸。记住，Resolution（分辨率）框是无关的（它指的是输出分辨率，它在 Web 上不是由设计人员或 Photoshop 决定的，而是由访问者的显示器决定的）。相反，图像的尺寸是由像素的数量决定的。需要选择 Resample Image（重新取样）框，这样才能改变图像的尺寸（而不是它的输出分辨率）。

提示　减小图像尺寸的另一个好办法是将图像上不需要的区域裁剪掉。

5.10　为网站添加图标

出现在地址栏、标签页和书签上小图标称为 **favicon**。这个词是 favorites icon（收藏夹图标）的简称。

苹果设备（iPhone、iPod touch 和 iPad）的主界面要求更大尺寸的图标，推荐尺寸为 114 像素 × 114 像素。不要为添加炫丽的圆角、阴影和反射光泽感到担心，这些设备的操作系统会自动进行处理。Android 操作系统也支持这些图标。

为网站添加图标

(1) 创建一个 16 像素 × 16 像素的图像，并保存为 ICO 格式，如图 5.10.1 所示。也可以保存为 PNG 和 GIF 格式。

(2) 为触屏设备创建一个 114 像素 × 114 像素的图像，并保存为 PNG 格式。

(3) 在 HTML5 文档的 head 部分，输入 `<link rel="shortcut icon" href="`*`favicon.ico`*`" />`，其中 *favicon.ico* 是服务器上图标的名称和位置。如果图像为 PNG，则输入 `<link rel="icon" type="image/png" href="favicon.png" />`。如果图像为 GIF，则输入 `<link rel="icon" type="image/gif" href="favicon.gif" />`。

(4) 在 HTML5 文档的 head 部分，输入 `<link rel="apple-touchicon" href="`*`/apple-touch-icon.png`*`" />`，其中 *apple-touch-icon.png* 是服务器上图标的名称和位置。

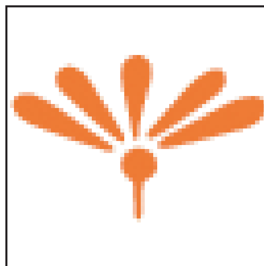


图 5.10.1　在现实环境中，favicon 比这里显示的要小。它们只有 16 像素 × 16 像素

提示　通常应将 favicon 保存为 ICO 格式（参见图 5.10.2 和图 5.10.3）。Photoshop 中有一个很有用的创建 ICO 格式图标的插件，它是由 Telegraphics 制作的（www.telegraphics.com.au/sw/）。

```
...
<head>
    <meta charset="utf-8" />
    <title>Farm Training Podcasts</title>
    <link rel="shortcut icon"
    → href="/favicon.ico" />
    <link rel="apple-touch-icon"
    → href="/apple-touch-icon.png" />
    ...
</head>
...
```

图 5.10.2　如果将文件命名为 favicon.ico 和 apple-touch-icon.png 并将它们放在网站的根目录下，即使没有这个 link 元素，大多数浏览器也会显示它们



图 5.10.3　favicon 通常用于地址栏、收藏夹和书签菜单、标签页。由于浏览器通常将图标显示在灰色或其他颜色的背景上，因此可能要将图标的背景改为透明的

提 示　也可以将 favicon 保存为 PNG 和 GIF 格式。一定要在 type 中使用正确的 MIME[①] 类型。

提 示　Internet Explorer 原来要求将 favicon.ico 放在网站的根目录。现在已经没有这个要求了，不过，如果没有 link 元素，浏览器仍会查看这个位置（参见图 5.10.4）。



图 5.10.4　在 iOS 设备中，将网站从 Safari 添加到主屏幕上时，就会用到 apple-touch-icon

① MIME 即 Multipurpose Internet Mail Extension（多用途互联网邮件扩展），是用于指定内容数据类型的互联网标准。——译者注

第 6 章

链　　接

本章内容

- ❑ 链接剖析
- ❑ 创建指向另一个网页的链接
- ❑ 创建锚
- ❑ 链接到特定的锚
- ❑ 创建其他类型的链接

链接是万维网的命脉。没有它们，每个页面都只能独立存在，同其他所有页面完全地分开。

6.1 链接剖析

链接有两个主要的部分：目标和标签。可以说，第一个部分目标（destination）是最重要的部分。使用它指定访问者点击链接时会发生什么。可以创建链接进入另一个页面，在页面内跳转，显示图像，下载文件，发送电子邮件，等等。不过，最常见的是连接到其他网页的链接（参见图 6.1.1），以及连接到其他网页特定位置（称为锚，anchor）的链接。目标是通过编写 URL（参见第 1 章）定义的，通常只能在（桌面）浏览器的状态栏中看到。

链接的第二个部分是标签（label），即访问者在浏览器中看到或在屏幕阅读器中听到，通过激活可以到达目标的部分。它可以是文本、图像或二者兼有。对于标签的文本，浏览器通常默认显示为带下划线的蓝色文字。通过 CSS 可以很容易地改变这一样式。

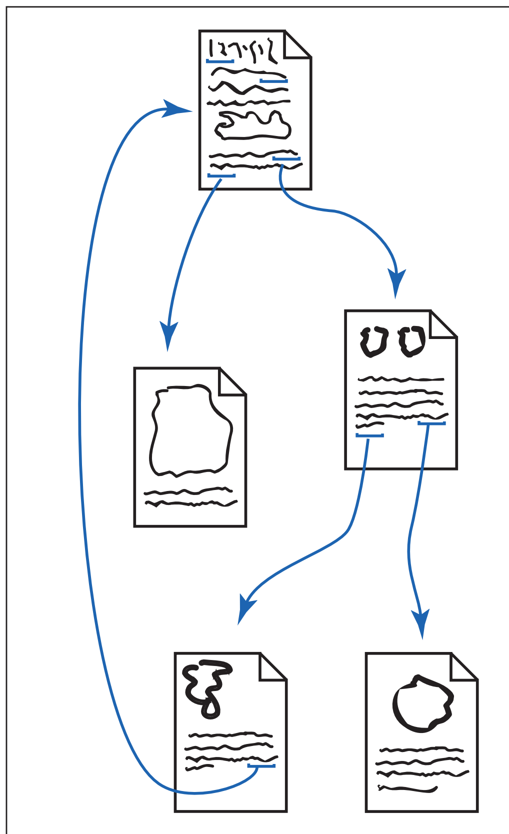


图 6.1.1　有的页面有很多指向其他页面的链接。有的页面只有一个链接。也有的页面没有任何链接

注意，人们常说用户点击链接，好像在对页面进行导航时鼠标是无处不在的。不过，我会尽量避免这个词，改用触发、激活，以反映用户与链接交互的方式的多样性。例如，触屏设备（如大多数智能手机和平板电脑）用户通过轻触激活链接，而其他移动用户可能通过轨迹球激活链接。同时，有的用户用键盘对页面进行导航，无论是出于偏好还是出于需要。他们可能有一些肢体障碍，使用鼠标、数码笔或类似的设备有一定困难或完全无法做到。这些用户通常通过 Tab 键找到链接（使用 Tab 键向前移动，使用 Shift + Tab 键向后移动），再通过 Enter 或 Return 键触发链接。

6.2　创建指向另一个网页的链接

如果你有多个网页，那么很可能希望创建从一个页面到另一个页面（以及返回）的链接（参见图 6.2.1 ～图 6.2.4）。还可以链接到其他网站的页面，无论是你自己制作的还是由他人创建的（参见图 6.2.5 ～图 6.2.8）。

1. 创建指向另一个网页的链接

(1) 输入 ，其中 *page.html* 是目标网页的 URL，如图 6.2.1 所示。

(2) 输入标签文本，也就是默认突出显示的文本，如图 6.2.2 所示，访问者激活它时，就会转到第 (1) 步中所指的页面。也可以添加一个 img 元素替代文本（或同文本一起）作为标签。（参见图 6.5.1，以及 6.5 节中的“将缩略图链接到图像”。）

(3) 输入 结束对链接的定义。

```
...
<body>

<article>
    <h1>Cookie and Woody</h1>

    <img src="img/cookiefora.jpg" width="143"
    → height="131" alt="Cookie" />

    <img src="img/woodygran.jpg" width="202"
    → height="131" alt="Woody" />

    <p>Generally considered the sweetest
    → and yet most independent cats in the
    → <a href="pioneer-valley.html">
    → Pioneer Valley</a>, Cookie and Woody
    → are consistently underestimated by
    → their humble humans.</p>
</article>

</body>
</html>
```

图 6.2.1　由于在 href 属性中只有文件名（没有路径），文件 pioneer-valley.html 必须与包含这个链接的网页位于同一个目录。否则，当用户激活该链接时，浏览器将找不到 pioneer-valley.html

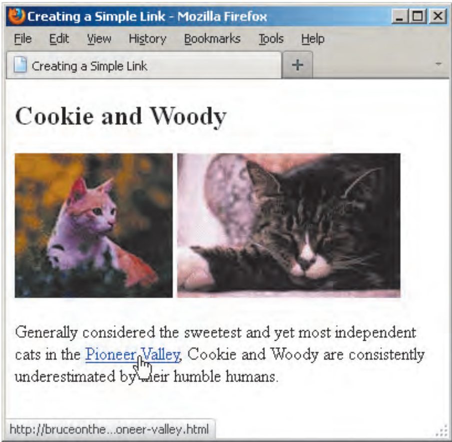


图 6.2.2　当访问者指向链接（在大多数浏览器中，默认显示为带下划线的蓝色文字）时，目标 URL 会显示在状态栏中。可以通过 CSS 改变这个默认样式。如果用户激活该链接……（参见图 6.2.3）

6

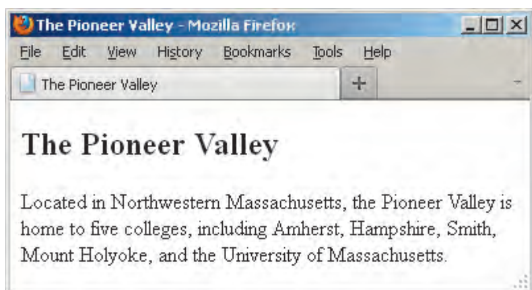


图 6.2.3　……与这个目标 URL 相关联的页面就会显示在用户的浏览器中

2. HTML5 的块级链接

HTML5 几乎允许在链接内包含任何类型的元素或元素组（参见图 6.2.4）。例如段落、列表、整篇文章和区块——几乎任何元素都行，交互式的内容除外，如其他链接、audio、video、表单元素、iframe 等。通过使用 HTML 验证器对页面进行测试（参见 20.5 节）以防止链接中出现不允许包含的元素。

```
...
<body>

<a href="giraffe-escapes.html">
    <hgroup>
        <h1>Giraffe Escapes from Zoo</h1>
        <h2>Animals worldwide rejoice</h2>
    </hgroup>
</a>

...
</body>
</html>
```

图 6.2.4　这种类型的链接在以前的 HTML 版本中是不允许的，但 HTML5 允许这样做

这些块级链接（block-level link，非官方称谓）是同 HTML 早期版本有巨大差异的地方。在以前的 HTML 中，链接中只能包含文本、图像等所谓的行内元素，也即标记文本短语（如 em、strong、cite 等）的元素（在 HTML5 中属于短语内容）。

有趣的是，尽管在以前的 HTML 规范中块级链接是不允许的，但浏览器都支持。这意味着你现在就可以使用它们，而且它们在旧的浏览器和现代浏览器中均能正常工作。不过，使用它们的时候也要小心（参见图 6.2.4 和图 6.2.5）。

```
...
<body>

<a href="pioneer-valley.html">
    <article>
        <h1>Cookie and Woody</h1>

        <img src="img/cookiefora.jpg" width=
        ➝"143" height="131" alt="Cookie" />

        <img src="img/woodygran.jpg" width=
        ➝"202" height="131" alt="Woody" />

        <p>Generally considered the sweetest
        ➝and yet most independent cats
        ➝in the Pioneer Valley, Cookie
        ➝and Woody are consistently
        ➝underestimated by their humble
        ➝humans.</p>
    </article>
</a>

...
</body>
</html>
```

图 6.2.5　不过做过了头。要避免这里显示的情况，链接包含了一大块内容。尽管这个链接仍然起作用，在 HTML5 里也是合法的，但屏幕阅读器可能会反复朗读全部内容，即使很多链接信息不是访问者通常想听到的。最好让链接只包含最相关的内容

有一些可访问性方面的注意事项，特别涉及不同的屏幕阅读器如何处理块级链接的问题。无障碍访问专家 Derek Featherstone 和 Steve Faulkner 的文章深入探讨了这个问题，

参见 http://simplyaccessible.com/article/html5-block-links/ 和 www.paciellogroup.com/blog/2011/06/html5-accessibilitychops-block-links/。他们建议将最相关的内容放在链接的开头，而且不要在一个链接中放入过多内容。Featherstone 指出，随着屏幕阅读器和浏览器逐渐开始官方支持块级链接，可访问性问题可能只是暂时的。

一般来说，用的最多的还是第一个例子（图 6.1.1）那样简单、传统的链接样式，不过也要知道，制作精巧的块级链接也是用得上的。

提示 href 指 hypertext reference（超文本引用）。

提示 可以改变标签文本的默认样式（参见第 10 章），甚至可以用图像作为标签（参见 6.5 节）。

提示 通常，对指向站内网页的链接使用相对 URL，对指向其他网站页面的链接使用绝对 URL。更多信息参见 1.7 节。

提示 指向另一个网站的页面的链接可能是这样的：**Label text**（参见图 6.2.6、图 6.2.7 和图 6.2.8）。rel 属性是可选的，即便没有它，链接也能照常工作。它描述包含链接的页面和链接指向的页面之间的关系。它也是另一种提升 HTML 语义化程度的方式。搜索引擎也可以利用这些信息。下面的地址维护了一个持续更新的 rel 值的列表：http://microformats.org/wiki/existing-rel-values。

```
...
<body>

<article>
    <h1>The Glory of Cats</h1>

    <p><a href="http://en.wikipedia.org/
    → wiki/Cat" rel="external"
    title="Cat
    → entry on Wikipedia">Cats</a> are
    → wonderful companions. Whether it's a
    → bottle cap, long string, or your legs,
    → they always find something to chase
    → around.</p>

    <p>In fact, cats are so great they even
    → have <a href="http://www.catsthe
    → musical.com/" rel="external"
    → title="Official site of Andrew
    → Lloyd Webber's musical">their own
    → musical</a>. It was inspired by T.S.
    → Eliot's <cite>Old Possum's Book of
    → Practical Cats</cite>.</p>
</article>

</body>
</html>
```

图 6.2.6 如果创建指向其他网站的链接，需要使用由 http://、服务器、完整路径和文件名构成的绝对 URL。rel 和 title 属性是可选的，但推荐使用 rel="external" 指示这是一个指向其他网站的链接（关于 cite 元素，参见第 4 章）

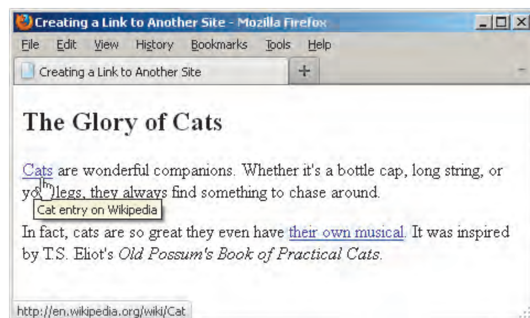


图 6.2.7 同指向站点内页面的链接一样，当访问者指向一个指向其他网站的链接（默认显示为带下划线的蓝色文字）时，目标 URL 会显示在状态栏，title 文本（如果有的话）显示在链接旁。如果访问者激活该链接（参见图 6.2.8）……



图 6.2.8 ……与这个目标 URL 相关联的页面就会显示在访问者的浏览器中

提示　仅指定路径，省略文件名，就可以创建指向对应目录下默认文件（通常为 index.html）的链接：**www.site.com/directory/**。如果连路径也省略，就指向网站的默认（首）页：**www.site.com**。

提示　在 URL 中应该全部使用小写字母，除非指向的页面或目录名称中含有大写字母。（对于你自己的网站，应使用小写字母对所有的文件夹和文件命名，并与链接 URL 对应。）

提示　不要让链接的标签太长。如果标签是句子的一部分，应在链接定义中仅保留关键词，让句子的其他部分位于 a 元素的外面。

提示　一定要在网站的每个页面包含指向网站各主要板块（包括首页）的导航。这可以让访问者自由地浏览网站，不管他们是直接访问网站，还是通过其他网站的链接访问的。你无法知道访问者会从哪里进入你的网站，可能是通过指向网站内页的"深度"链接，因此你应该让他们可以从那里开始访问网站的其余部分。

提示　不管怎样，应避免使用"点击此处"作为标签。这种类型的链接文本在万维网上实在是太常见了，它会破坏网站的可用性和可访问性，从而对网站的拥有者产生不利的影响。当用户快速扫过页面上的链接（无论是通过屏幕还是通过屏幕阅读器）时，会发现"点击此处"缺乏上下文（"点击此处？为什么？"）。它对激活链接几乎不会产生激励，而且依赖于访问者阅读链接周围的文字，并寄希望于这些文字可以解释链接的目的。不难理解，访问者通常更倾向于跳过这样的链接。此外，正如本章开头提到的，"点击"一词并不适用于用户触发链接的所有方式。相反，应该使用文本中已经存在的关键词对链接进行标识。例如，应使用"了解我们的销售情况"，而不是"点击此处了解我们的销售情况"。

提示　要创建指向某页面特定位置的链接，可以使用锚（参见 6.3 节）。

提示　正如本章引言中提到的，可以使用键盘浏览网页。每次按下 Tab，焦点就会转到 HTML 代码中出现的下一个链接、表单控件或图像映射。这个顺序不一定与屏幕上出现的顺序一致，因为页面的 CSS 布局可能不同。通过使用 HTML 的 `tabindex` 属性，可以改变使用 Tab 键的顺序，但不建议使用它，因为它在大多数情况下是不必要的、过时的做法。（在某些情况下它可能有用，但通常这是出于使用 JavaScript 增强交互体验的需要，这已属于高级主题。）相反，应在对内容进行标记时稍加注意，确保使用 Tab 键的顺序是符合逻辑的。应对自己的页面使用 Tab 键进行测试，就像用户的操作一样，再对 HTML 作相应的调整。

target 属性

可以让链接在新的窗口或标签页（取决于所用的浏览器）打开，但这被认为是一种不好的实践，不推荐使用。这样说是有理由的。

首先，应该让用户决定是否在不同的窗口或标签页打开链接，而不是让 HTML 开发人员决定。否则，我们就支配了用户的浏览行为。

也有一些可用性和可访问性方面的考虑。缺乏经验的用户在激活一个链接却没有在当前窗口看到结果时可能会有一些疑惑。使用浏览器并非对所有人都是一件容易的事；我曾向不同年龄段的人展示标签页，他们之前并不知道可以同时打开多个页面。类似地，屏幕阅读器等辅助设备的用户将不得不另外花费努力去转向新的窗口或标签页，前提还是他们明确知道是哪个窗口或标签页加载了新的内容。

如果所有这些并未说服你避免使用在新窗口和标签页打开的链接，或者你的老板或客户不赞同你不使用它们的理由，可以看看如何创建这种链接：在链接定义中输入 `target="`*`window`*`"`，其中 *`window`* 是应该显示相应页面的窗口的名称（由你自己选择）。

例如，`<a href="some-page.html" target="doodad">Some page</a>` 会在名为 doodad 的新窗口或标签页中打开 `some-page.html`。

如果让多个链接指向同一个窗口（即使用同一个名称），链接将都在同一个窗口打开。或者，如果你希望链接总是在不同的窗口或标签页打开（即使多次激活同一个链接），就使用 HTML 预定义的名称 `_blank`，即使用 `target="_blank"`。

不过我还是不推荐你这样做，能免则免。

`target` 还有一种用法，就是在 `iframe` 中打开链接。可以用同样的方法编写 `target`，只是其值应与 `iframe` 的 `id` 对应。你很少有机会用到这个，特别是在通常不推荐使用 `iframe` 的情况下（尽管它们有时会有用武之地）。关于 `iframe` 元素的更多信息，参见 https://developer.mozilla.org/en/HTML/Element/iframe。

尽管这里不会讲解图像映射的用法，但你应知道它们是用来为单张图片的一个或多个区域添加链接的。可以将每个链接区域定义为矩形、圆形或多边形。缺乏经验的编码人员常常误用它们创建基于图像的导航，而不是使用更为合理的技术，如用 CSS 或图像替换技术（如果 CSS 不够用的话）制作的 HTML 文本。图像映射的鼎盛已是多年以前的事情了，那时上述这些技术还不普遍（甚至不可能实现）。现在图像映射已经用得很少了，但偶尔还有一些合理的用例，例如，对于某个国家的地图，希望定义指向各地区、省或州的链接。可以在网上搜索“HTML image maps”（HTML 图像映射）了解有关图像映射的更多信息。

6.3 创建锚

通常，激活一个链接会将用户带到对应网页的顶端。如果要想用户跳至网页的特定区域，可以创建一个锚，并在链接中引用该锚，参见图 6.3.1 和图 6.3.2。

```
...
<body>

<h1>Frankie and Johnny</h1>

<header>
    <h2>Table of Contents</h2>
    <nav>
       <ul>
          <li><a href="#intro">Introduction</a></li>
          <li><a href="#main-characters">Description of the Main Characters</a></li>
          <li><a href="#rising-action">Rising Action</a></li>
       </ul>
    </nav>
</header>

<article>
    <h2 id="intro">Introduction</h2>
    <p>This is the intro. If I could think of enough things to write about, it could span a few
    ➝ pages, giving all the introductory information that an introduction should introduce.</p>

    <h2 id="main-characters">Description of the Main Characters</h2>
    <p>Frankie and Johnny are the main characters. She’s jealous, and seems to have a reason to be.
    ➝ He’s a sleaze, and will pay the price.</p>

    <h2 id="rising-action">Rising Action</h2>
    <p>This is where everything starts happening. Johnny goes out, without Frankie, without even
    ➝ tellin’  her where he's going. She's not crazy about it, but she lets him go. A while later,
    ➝ she gets thirsty and decides to go down to the corner bar for some beer. Chatting with the
    ➝ bartender, she learns that Johnny has been there with no other than Nellie Bly. Furious, she
    ➝ catches the crosstown bus to find him.</p>
</article>

</body>
</html>
```

图 6.3.1　每个以 # 开头的链接 href 值都指向拥有相应 id（不含 #）的元素。例如，Rising Action 指向 <h2 **id="rising-action"**>Rising Action</h2>。可以为任何元素指定 id，只要任何给定的 id 在一个页面中只存在一次（参见 3.15 节）。这个例子还让你提前看到了无序列表（ul）的应用。无序列表是目前万维网上使用频率最高的列表类型（第 15 章将深入讨论列表）

```
...
<header>
    <h2>Table of Contents</h2>
    <nav>
        <ul>
            <li><a href="#intro">
            ➝ Introduction</a></li>
            ...
        </ul>
    </nav>
</header>

<article>
    <section id="intro">
        <h2>Introduction</h2>
        <p>This is the intro...</p>
    </section>

    <section id="main-characters">
        <h2>Description of the Main
        ➝ Characters</h2>
        ...
    </section>

    <section id="rising-action">
        <h2>Rising Action</h2>
        ...
    </section>
</article>

</body>
</html>
```

图 6.3.2 第一个例子（图 6.3.1）设计得很简单，只是为了演示锚的基本用法。不过，可以将每个答案放入一个 section 元素以进一步增强语义，同时将 id 放在这些元素内，而不是放在标题里。这这些内容的方法（未在让它们成为 article 父元素的区块。另一种标记这里演示）是将每个答案视为单独的 article，这可以通过移除 article 父元素，并将每个 section 改为 article 实现。这取决于你希望以哪种方式描述你的内容含义（你认为这些答案是一篇文章还是各自独立的文章）

创建锚的步骤

(1) 将光标放在希望用户跳转至的元素的开始标记里。

(2) 输入 id="*anchor-name*"，其中 *anchor-name* 是在内部用来标识网页中这部分内容的文字。一定要在元素名称和 id 之间保留一个空格，例如 <h2 **id="rising"**>。

提示 为锚 id 赋予有意义的名称，增强 HTML 文档的语义丰富度。换句话说，避免使用 anchor1、item5 这样的 id。

提示 id 中不能使用空格，应该用短横线分隔不同的单词。

提示 在某些情况下，可能需要在每个内容区块的下方包含一个指回目录的链接（你可能已经习惯于看到它们了，它们通常显示为“返回顶部”）。不过，如果你的页面有好几个长的区块，可以考虑将它分成多个页面。

6.4 链接到特定的锚

通过 id 创建了锚以后，就可以定义一个链接，当用户触发它时，页面会直接跳至文档中包含该锚的区域（参见图 6.4.1 和图 6.4.2），而不是文档的顶端。

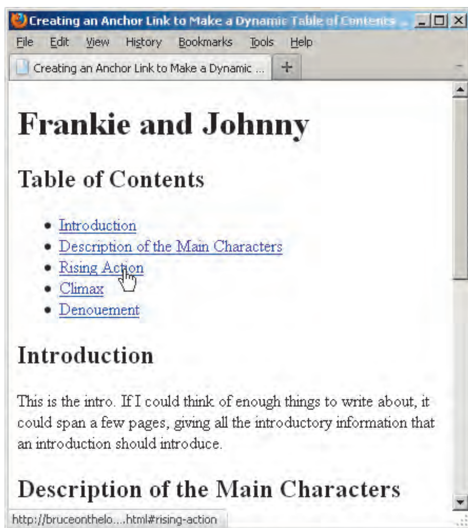


图 6.4.1 在桌面浏览器中，当访问者将鼠标指向引用锚的链接时，URL 和锚名称会显示在状态栏中（在窗口的左下角）

6

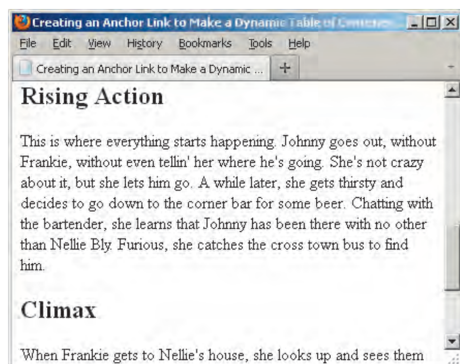


图 6.4.2　当访问者激活该链接时，锚引用的页面特定部分就会显示在浏览器窗口的顶部

创建链接到特定锚的链接

(1) 输入 ，其中 *anchor-name* 是目标的 id 属性值（参见 6.3 节“创建锚的步骤”中的第 (2) 步）。

(2) 输入标签文本，即突出显示的文本（通常默认显示为带下划线的蓝色文字），也即用户激活它时将用户带到第 (1) 步中引用的区域的文本。

(3) 输入 结束对链接的定义。

提 示　如果锚位于另一个文档，就使用 <a href ="***page.html#anchor-name***"> 引用该区域。（在 URL 和 # 之间没有空格。）如果锚位于另一台服务器上的页面，则需输入 <a href="***http://www.site.com/path/page.html#anchor-name***">（没有空格）。

提 示　我们显然无法在别人的页面中添加锚，但可以利用他们已经做好的锚。查看他们文档的源代码，就可以找到某区域对应的锚（关于查看源代码的方法，参见 2.8 节）。

提 示　如果锚位于页面的底部，那么它可能不会显示在窗口的顶部，而是显示在中间。

6.5　创建其他类型的链接

并非只能创建指向其他网页的链接，其实可以创建指向任何 URL 的链接——RSS 源、希望访问者可以下载的文件、电子邮件，等等（参见图 6.5.1）。

创建其他类型链接的步骤

(1) 输入 <a href="。

(2) 输入 URL。

(3) 对于指向万维网上任何文件（包括图像、ZIP 文件、程序、PDF、Excel 电子表格等）的链接，输入 http://*www.site.com/path/file.ext*，其中 *www.site.com* 是服务器的名称，*path/file.ext* 是目标文件（含扩展名）的路径。

(4) 输入 ">。

(5) 输入链接的标签。标签是默认以下划线和蓝色突出显示的文本，当用户激活它时，会将访问者带到第 (2) 步中引用的 URL。也可以添加一个 img 元素替代文本（或同文本一起）作为标签（参见图 6.5.1 以及本节末尾的“将缩略图链接到图像”）。

提 示　如果链接指向的文件是浏览器不知道如何处理的类型（例如 Excel 文件），浏览器将试着打开一个辅助程序来查看这个文件，或试着将它下载到访问者的磁盘上。

提 示　对供访问者下载的大文件和文件组进行压缩是个好办法。例如，一套保存为 PSD 文件的 Photoshop 模板。在网上搜索“ZIP and RAR”，查找创建和打开使用这些流行的压缩格式的文件存档的工具。

```
...
<body>

<h1>Other Types of Links</h1>

<p>There are lots of different kinds of links that you can create on a Web page. More precisely,
→ there are a lot of different files you can link to on your Web page.</p>

<p>You can create links directly to <a href="img/blueflax.jpg">a photo</a> or even make links out
→ of photos.</p>

<p>For example, here are Cookie and Woody again, except this time they are linked to other pages.
→ <a href="cookie.html" title="All about Cookie"><img src="img/cookiefora.jpg" width="143"
→ height="131" alt="Cookie" /></a> <a href="woody.html" title="All about Woody"><img
src="img/
→ woodygran.jpg" width="202" height="131" alt="Woody" /></a></p>

<p>You can link directly to <a href="http://www.sarahsnotecards.com/catalunyalive/segadors.
mov"
→ rel="external">a video</a> file, too, though it's usually better to link to a page with the
→ video embedded in it, such as with the <abbr title="Hypertext Markup Language revision 5">HTML5
→ </abbr> <code>video</code> element.</p>

<p>Although you can make a link to <a href="mailto:someone@somedomain.com">someone's email
→ addresss</a> with the <code>mailto:</code> protocol, I don't recommend it, since spambots pick
→ those up and then bombard them with spam. It's too bad, because they are so convenient. If you
→ activate the link, it opens your email program. It's probably better to offer your email address
→ in a descriptive way, like &quot;someone at somedomain,&quot; although that isn't always foolproof
→ either.</p>

<body>
</html>
```

图 6.5.1 可以创建指向各种类型 URL 的链接。这个页面包含五个链接，但两个包围图像的链接可能并不对所有浏览器可见（参见图 6.5.2）。（这个例子还频繁使用 abbr 元素标记缩写词，以及指示内容为代码的 code 元素。这两个元素均在第 4 章中讲到过。）

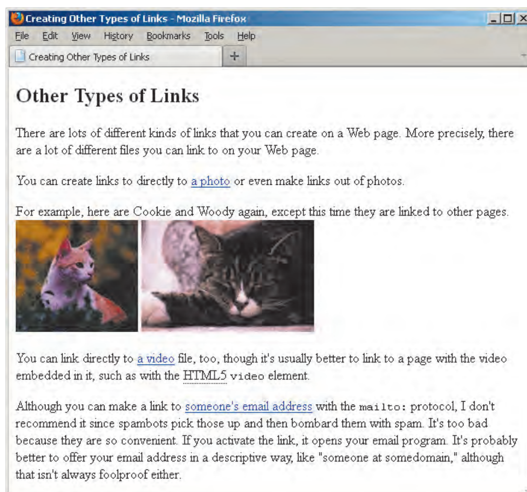


图 6.5.2 无论链接通向哪里，它们在浏览器中的默认样式都是一样的，除非其中包含图像（有的浏览器在图像链接周围添加边框，有的则不会）。注意，我尽量创建与文本主体混在一起的标签，而不是使用“点击此处”（Click me）作为标签

提示 尽管可以链接到 PDF 和其他非 HTML 文档（Word、Excel 等），但应尽量避免这样做。相反，应链接到包含有关信息的 HTML 页面。PDF 可能要花更长的时间加载，而且有的浏览器和系统（特别是旧的浏览器和系统）在尝试显示它们的时候会变得很慢。当 PDF 是唯一的选择时，要让用户知道链接指向的是 PDF 而不是另一个 HTML 页面，以免让他们感到意外（如果让用户进入耗时的下载，他们是不会领情的）。这条建议对其他非 HTML 文档也适用。显示给用户的信息可以是简单地用括号包围的文件类型和大小，也可以显示一个图标。例如（不包含图标）：`<a href="q2-sales-report.pdf">Q2 Sales Report **(PDF, 725kb)**</a>`。还可以在链接中包含 `title` 属性（如 `title="Opens a PDF"`），特别是在当你希望把括号说明放在链接外的时候。

提示 如果要“创建指向 iTunes Store、App Store、iBookstore 和 Mac App Store 上内容的链接”（链接如下），可以使用苹果公司的 Link Maker（http://itunes.apple.com/linkmaker）生成 URL 放在 HTML 里。如果你是联盟成员（www.apple.com/itunes/affiliates/），人们通过你的链接购买物品时，苹果公司会为你支付佣金。

将缩略图链接到图像

你肯定访问过显示几个缩略图（图像的微缩版本）的相册页面，其中缩略图链接到大一些的图像。这样就可以一次看到很多图片，让访问者选择要查看全尺寸版本的图像。

实现其基本版本类似于将 Cookie 和 Woody 图像链接到其他页面的示例代码（参见图 6.5.1）。这些页面中的每一个都包含一个全尺寸照片。（也可以在单独一个动态页面中实现，但这是超出 HTML 能力的高级用法。）

注意不要让任何给定页面上的缩略图数量太多。它们可能很小，但每个缩略图都会生成对 Web 服务器的独立请求，合在一起就会让页面变慢。没有规则规定一个页面放多少缩略图是合适的。这部分取决于页面加载的其他资源的数量和大小，也取决于网站的目标受众。例如，移动设备加载资源通常要慢一些。

因此，如果你有很多缩略图，就要考虑将它们分入多个页面。通常，每页 20 ~ 30 个缩略图是比较合理的，同时也要再次对上述因素进行考虑。可以对页面进行测试，再确定最合适的数量。

最后，推荐使用无序列表（`ul`，将在第 15 章讲到）对缩略图列表进行标记。

第 7 章

CSS 构造块

7

本章内容

- ❑ 构造样式规则
- ❑ 为样式规则添加注释
- ❑ 层叠：当样式发生冲突时
- ❑ 属性的值

HTML 定义内容的含义，为网页构建基本的结构，而 CSS（Cascading Style Sheet，层叠样式表）则定义它们的外观。

样式表不过是一种文本文件，其中包含一个或多个（通过属性和值）决定网页某特定元素如何显示的规则。CSS 里有控制基本格式（如字号和颜色）的属性，有控制布局（如定位和浮动）的属性，还有决定访问者打印时在哪里换页的打印控制元素。CSS 还有很多控制项目显示或消失的动态属性，可以用于创建下拉列表和其他交互性组件。

CSS2 是新旧浏览器支持最为广泛的版本，因此本书将大量讨论这一版本的内容。CSS3 目前还没有成为规范，它以 CSS2 为基础，提供了大量设计人员和开发人员长期期待的功能。值得庆幸的是现代浏览器已经实现了一些 CSS3 的组件，因此从现在起你就可以用它们了。本书将讲解其中一些最为有用且浏览器支持情况良好的功能。

CSS 很棒的一点在于开发人员可以在网页之外创建它，再将它同时应用于网站上所有的页面。它灵活、强大而且高效，可以为你节省大量的时间和带宽。

为了充分利用 CSS 的优势，必须依照 HTML 相关章节的推荐做法对网页进行良好并一致地标记。

7.1 构造样式规则

样式表中的每条规则都有两个主要部分：选择器（selector）和声明块（declaration block）。选择器决定哪些元素受到影响；声明块由一个或多个属性/值对（每个属性/值对构成一条声明）组成，它们指定应该做什么（参见图 7.1.1 和图 7.1.2）。

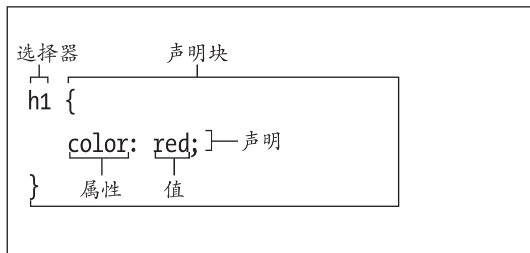


图 7.1.1 样式规则由选择器（表示哪些元素将进行格式化）和声明块（描述要执行的格式化）组成。声明块内的每条声明都是一个由冒号隔开、以分号结尾的属性/值对。声明块以前花括号开始，以后花括号结束

```
h1 {
    background: yellow;
    color: red;
}
```

两条声明，每条都有一个属性和一个值

图 7.1.2　声明的顺序并不重要，除非对相同的属性定义了两次。在这个例子中，color: red 也可以放在 background: yellow 前面，效果是一样的。注意额外的空格和缩进（可选，但推荐包含）提高了可读性

构造样式规则的步骤

(1) 输入 *selector*，这里的 *selector* 表示希望进行格式化的元素。第 9 章将讲解如何创建各种类型的选择器。

(2) 输入 {（前花括号）开始声明块。

(3) 输入 *property:value*;，其中 *property* 是 CSS 属性的名称，描述要应用哪种格式；*value* 是该属性允许的选项之一。本书从第 8 章开始讲解 CSS 属性和值。

(4) 根据需要，重复第 (3) 步。通常一行输入一个 property: value（一条声明）。

(5) 输入 }，结束声明块和样式规则。

提 示　在样式规则中可以添加额外的空格、制表符或回车，从而提高样式表的可读性（参见图 7.1.2）。示例中的格式或许是编码人员中最为常见的一种格式。

提 示　每组属性/值对都应该使用一个分号与下一组属性/值对分开，但列表中最后一对后面的分号可以省略。不过，推荐包含这个分号，这被证明是一种最佳实践。

提 示　缺少（或重复）分号会使浏览器忽略样式规则。

7.2　为样式规则添加注释

在 CSS 中添加注释是个好主意，这样就可以标注样式表的主要区域，或者只是对某条规则或声明的说明。注释不仅对你有用，对阅读代码的其他人也有好处。当你过几个月再看你写的代码时，会庆幸自己留下了这些注释。

为样式规则添加注释

(1) 在样式表中，输入 /* 开始注释。

(2) 输入注释。

(3) 输入 */ 结束注释。

提 示　注释可以包含回车，因此可以跨越多行（参见图 7.2.1）。

```
/*这是一段CSS注释。它可以只有一行，也可以
→ 跨越多行。这个注释比大多数注释都长。CSS注
→ 释不会同网站HTML内容一起显示在浏览器中。
→ 下面一条注释更合乎注释的常规用法。 */

/*为旧的浏览器设置特定HTML5元素的默认呈现样式。*/

article, aside, details, figcaption, figure,
→ footer, header, hgroup, menu, nav, section {
    display: block;
}
```

图 7.2.1　注释可长可短，但它们通常较短。可以根据你的想法，用它们描述样式规则或一组相关的规则的目的。注释做得好可以让样式表更易于维护

提 示　不能将注释放在另一个注释里。换句话说，注释中不能包含 */。

提 示　可以将注释放在单独的行上（参见图 7.2.1），也可以放在声明块里（参见图 7.2.2），也可以放在规则后面（参见图 7.2.2）。

```
/* Add rounded corners in supporting browsers */
.box {
    -webkit-border-radius: 12px; /* Safari 3-4 */
    -moz-border-radius: 12px; /* Firefox 3.6 及以下 */
    border-radius: 12px; /* 现代浏览器 */
} /* 再写一条注释*/
```

图 7.2.2 也可以在声明块里或规则后面插入注释

提示 注释是很有用的组织工具。样式表很快就会变得很长，因此，对样式表进行组织对于保持 CSS 易于维护是至关重要的。通常，将相关的规则放在一起，形成分组，并在每组前面放置一段描述性的注释，参见图 7.2.3。

```
/* GLOBAL NAVIGATION（全站导航）
------------------------------- */
……全站导航的样式规则……


/* MAIN CONTENT（主体内容）
------------------------------- */
……主体内容的样式规则……


/* SIGN-UP FORM（注册表单）
------------------------------- */
……注册表单的样式规则……


/* PAGE FOOTER（页脚）
------------------------------- */
……页脚的样式规则……
```

图 7.2.3 注释可以让样式表管理变得很轻松。对样式表中的主要区域添加注释，就可以保持样式表井然有序。下面的格式（使用大写字母和一条下划线）可以很清楚地标识分组的开始位置。这种做法可以很清晰地将它们与其他注释（如图 7.2.1 和图 7.2.2 中的注释）区分开来

提示 无论你采用怎样的注释格式，推荐你定下一种并坚持使用，特别是在团队协作的时候。

提示 可以将注释放在样式规则的周围或里面，从而对浏览器隐藏样式规则（参见图 7.2.4）。这是对样式表进行测试的一种好方法，不必永久性地删除注释部分，直到你认为可以删除它们了。这也是很有用的调试工具，可以将你认为可能引起问题的地方“注释掉”，再在浏览器中刷新页面，查看问题是不是解决了。

```
img {
    border: 4px solid red;
    /* margin-right: 12px; */
}
```

图 7.2.4 可以将你不希望对页面产生影响的声明“注释掉”。这里，所有的图像都会拥有一条 4 像素宽的红色实线边框，但不会有右侧外边距，因为 `margin-right: 12px;` 位于一条注释里面。注释也可以包围整个规则，只要这个注释里面没有任何注释

提示 出于演示的目的，这些例子都使用了很多注释，但不要因此认为要对所有内容添加注释。如果样式表里的注释过多，反而会难以阅读。要根据需要混合使用组织性注释和描述性注释。要针对你和团队里的其他成员，找到一个平衡点。

7.3 层叠：当规则发生冲突时

很多地方都可以应用样式。正如你在第 1 章中学到的，每个浏览器都有其默认样式。不过，你可以用自己的样式覆盖它们或对它们进行补充。应用样式有三种方式：从一个

或多个外部文件导入（推荐），如图 7.3.1 中的代码所示，插入到 HTML 文档的顶部，或直接应用于代码中特定的元素上（不过，应该尽可能地避免这样做）。下一章将对这三种方式做具体讲解。

```
p {
    color: #36c;
    font-family: "Trebuchet MS",
    ➝ "Helvetica", sans-serif;
    font-weight: bold;
}

img {
    float: left;
    margin-right: 10px;
}
```

图 7.3.1　这是图 7.3.2 中 HTML 文档的样式表。现在不需要关心太多的细节，只需要意识到这里有一条关于 p 元素（而不是 h1、em 或 small 元素）的规则

同时，一些浏览器还允许访问者创建他们自己的样式表，并将其应用于他们所访问的页面（包括你的页面）。此外，有一些样式是由子元素继承自父元素的。

你可能会问，对某一给定元素应用多条样式规则时，会发生什么情况？CSS 用层叠的原则来考虑继承（inheritance）、特殊性（specificity）和位置（location）等重要特征，从而判断相互冲突的规则中哪个规则应该起作用。

我们从继承开始讨论。很多 CSS 属性不仅影响选择器所定义的元素，而且会被这些元素的后代继承（参见图 7.3.1、图 7.3.2 和图 7.3.3）。例如，假设要让所有的 h1 元素显示为蓝色，并带有红色边框。不过，color 属性是继承的，而 border 属性不是。那么，h1 元素里包含的任何元素都会是蓝色的，但不会有红色的边框。你将在描述每个属性的章节（以及附录 B）了解到哪些属性是继承的。另外，对大多数属性来说，还可以使用 inherit 值强制进行继承（参见 7.4 节）。

```
…
<body>

<h1>The Ephemeral Blue Flax</h1>

<img src="img/blueflax.jpg" width="300"
➝ height="175" alt="Blue Flax (Linum
➝ lewisii)" />

<p>I am continually <em>amazed</em> at the
➝ beautiful, delicate Blue Flax that somehow
➝ took hold in my garden. They are awash in
➝ color every morning, yet not a single
➝ flower remains by the afternoon. They are
➝ the very definition of ephemeral.</p>

<p><small>&copy; by Blue Flax Society.
➝ </small></p>

</body>
</html>
```

图 7.3.2　em 和 small 元素包含在 p 元素里，因此它们是 p 的子元素。不过，h1 并不是 p 的子元素，因此它不像其他文本一样是蓝色的，参见图 7.3.3



图 7.3.3　图 7.3.1 中并没有显式地指定 em 和 small 元素的样式，它们将从其父元素 p 那里继承字体、粗体和颜色。斜体来自浏览器为 em 设的默认样式。出于同样的原因，由 small（表示法律“细则”）标记的法律通告的字号要比普通文本的小。h1 本身没有设置样式，它也不是 p 的子元素，因此它完全根据浏览器的默认样式进行显示（另见彩插）

继承决定了一个元素没有应用任何样式时应该怎样显示，而特殊性则决定了应用多个规则时应该怎样显示（参见图 7.3.4、图 7.3.5 和图 7.3.6）。根据特殊性原则，选择器越特殊，规则就越强。讲得通，对吧？因此，如果一条规则声明所有的 h1 元素都应该是蓝色的，而另一条规则声明所有 class 为 spanish 的 h1 元素都应该是红色的，那么对于所有 class 为 spanish 的 h1 元素，第二条规则将覆盖第一条规则，因为 h1.spanish 是比 h1 更为特殊的选择器。

```
p {
    color: red;
}

p.group {
    color: blue;
}

p#last {
    color: green;
}

p#last {
    color: magenta;
}
```

图 7.3.4　在这个例子中，有四个具有不同特殊性的规则。第一个影响所有 p 元素，第二个只影响 class 为 group 的 p 元素，而第三个和第四个则只影响 id 为 last 的唯一的 p 元素

注意，id 属性被认为是最特殊的（因为它们在一个文件中必须是唯一的），而带 class 属性的选择器则比不带 class 的更特殊。同时，具有多个 class 的选择器比只有一个 class 的更特殊。在特殊性次序中，最低级的是只有元素名的选择器，这时继承的规则被认为是最一般的，可以被任何其他规则覆盖。

关于特殊性计算的精确规则，参见 CSS 规范 6.4.3 节（www.w3.org/TR/CSS21/cascade.html#specificity）。

```
...
<body>

<p>Here's a generic <code>p</code> element.
→ It will be red.</p>

<p class="group">Here's a <code>p</code>
→ element with a <code>class</code> of
→ <code>group</code>. There are two rules
→ that apply, but since the <code>p.group
→ </code> rule is more specific, this
→ paragraph will be blue.</p>

<p id="last" class="group">Here's a <code>
→ p</code> element with an <code>id</code>
→ of <code>intro</code>. There are four rules
→ that could apply to this paragraph. The
→ first two are overruled by the more
→ specific last two. The position breaks
→ the tie between the last two: the one
→ that appears later wins, and thus this
→ paragraph will be magenta.</p>

</body>
</html>
```

图 7.3.5　这里有三个段落：一个是一般的，一个是带有一个 class 的，一个是同时带有一个 class 和一个 id 的。

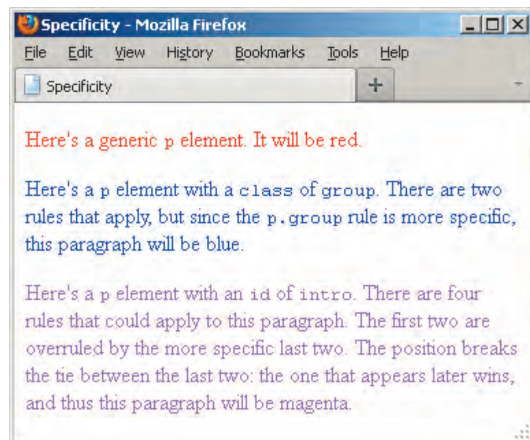


图 7.3.6　由于第三个和第四个规则具有相同的特殊性，位置就成为一个考虑因素。由于第四条规则最后出现，因此它的优先级更高

有时候，特殊性还不足以判断在相互冲突的规则中哪一个应该优先。在这种情况下，规则的位置就可以起到决定的作用：晚出现的优先级高（参见图 7.3.4、图 7.3.5 和图 7.3.6）。例如，直接应用在 HTML 元素上的规则（再说一遍，这种做法不推荐使用）被认为比外部样式表中或插在 HTML 文档顶部的特殊性相同的规则出现得更晚（因此优先级更高）。详细信息参见 8.5 节。

如果这还不够，那么还可以在某条规则的末尾加上 `!important`，以声明这个规则比其他规则更重要，超过整个优先级体系（除少数情况以外，这种做法也不推荐使用）。

总之，在没有规则的时候，很多样式都从父元素传递到子元素。当两个规则发生冲突时，规则更特殊的优先级（即重要性）更高（如果不考虑位置的话）。如果两个规则的特殊性相同，则晚出现的优先级更高。

如果感觉这些东西还是很费解，也不必担心。一旦你真正开始使用 CSS 以及不同的选择器，你会发现，在大多数情况下，层叠会像你预期的那样运行。

7.4 属性的值

每个 CSS 属性对于它可以接受哪些值都有不同的规定。有的属性只能接受预定义的值。有的属性接受数字、整数、相对值、百分数、URL 或者颜色。有的属性可以接受多种类型的值。每个属性可接受的值将在描述该属性的章节（大多在第 10 章和第 11 章）列出来，不过这里将讨论基本体系。

1. inherit

对于任何属性，如果希望显式地指出该属性的值与对应元素的父元素对该属性设定的值相同，就可以使用 `inherit` 值。

2. 预定义的值

大多数 CSS 属性都有一些可供使用的预定义值。例如，`float` 属性可被设为 `left`、`right` 或 `none`。与 HTML 不同，不需要（也不能）将预定义的值放在引号里，参见图 7.4.1。

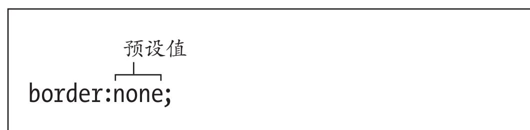


图 7.4.1 很多 CSS 属性只接受预定义的值。要准确地输入这些值，并且不要加上引号

3. 长度和百分数

很多 CSS 属性的值是长度。所有长度都必须包含数字和单位，并且它们之间没有空格。例如 `3em`、`10px`，参见图 7.4.2。唯一的例外是 `0`，它可以带单位也可以不带。



图 7.4.2 长度必须显式指出单位。在单位和数值之间应该没有空格

有的长度是相对于其他值的。一个 em 的长度大约与对应元素的字号相等，因此 `2em` 表示“字号的两倍”。（当 em 用于设置元素的 `font-size` 属性本身时，它的值继承自对应元素的父元素的字号。）ex 应与字体的 x 字高相等，也就是与这种字体中字母 x 的高度相等。不过，浏览器对 ex 的支持不是太好，因此很少会用到它。

像素（px）并不是相对于其他样式规则的。例如，以 px 为单位的值不会像以 em 为单位的值那样受 `font-size` 设置的影响。某种设备上一个像素的大小不一定与另一种设备上的完全相等。（参见 Peter-Paul Koch 对此

问题的详细说明，www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html。）

还有一些无需说明的绝对单位，如磅（pt），应该在为打印准备的样式表中保留这个单位。（此外还有一些，不过这里无需再提，因为它们在实践中很少用到。）一般来说，应该只在输出尺寸已知的情况下使用绝对长度（如在打印的页面中使用 pt）。

百分数（如 65%）的工作方式很像 em，它们都是相对于其他值的值，参见图 7.4.3。

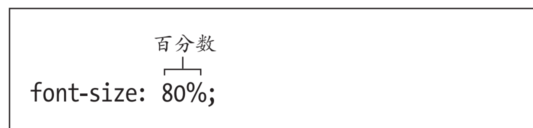


图 7.4.3 百分数通常是相对于父元素的。因此，在这个例子中，字号将被设为父元素字号的 80%

在上述单位中，最常使用的是 em、像素和百分数。

4. 纯数字

只有极少数的 CSS 属性接受不带单位的数字，如 3。其中最常见的就是 line-height（参见图 7.4.4）和 z-index（分别参见 10.6 节和 11.15 节）。（另外几个属性主要用于打印和听觉样式表，而且没有得到很好的支持。）

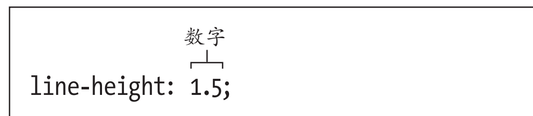


图 7.4.4 不要将数字、整数和长度弄混。数字或整数没有单位（如 px）。这个例子中的值将与字号相乘，得到行高

5. URL

有的 CSS 属性允许开发人员指定另一个文件（尤其是图像）的 URL。在这种情况下，使用 url(*file.ext*)，其中 *file.ext* 是目标资源的路径和文件名，参见图 7.4.5。注意，规范指出，相对 URL 应该相对于样式表的位置而不是相对于 HTML 文档的位置。

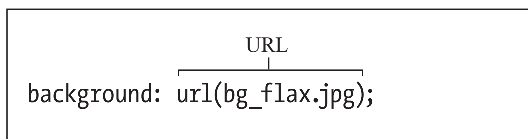


图 7.4.5 CSS 属性中的 URL 无需使用引号包围

可以在文件名上加上引号，但这不是必需的。此外，在单词 url 和前括号之间不应该有空格。括号和地址之间允许有空格，但这不是必需的（通常也不这样做）。

关于编写 URL 本身的信息，参见 1.7 节。

6. CSS 颜色

有多种方法为 CSS 属性指定颜色。首先（也是最容易的），可以使用预定义的颜色关键词作为颜色值。CSS3 指定了 16 个基本的名称（参见图 7.4.6），又增加了 131 个名称，从而组成了 147 种 SVG 1.0 颜色关键词。完整的列表见 www.w3.org/TR/css3-color/#svg-color。

当然，除了几样最简单的名称，没有人记得住这些颜色名。可以使用 Adobe Photoshop 等工具进行取色，但它们并不使用 CSS 颜色名。在实践中，更常见的定义 CSS 颜色的方法是使用十六进制格式（这是目前为止最为常见的方式）或 RGB 格式。后面将讲到，还可以使用 HSL 格式指定颜色，使用 RGBA 和 HSLA 指定颜色的透明度，这些都是 CSS3 中新增的方式。

- RGB

可以通过指定红、绿、蓝（这也是 RGB 这一名称的由来）的量来构建自己的颜色。可以使用百分数、0 ~ 255 之间的数字或十六进制数来指定这三种颜色的值。例如，如果创建一种深紫色，可以使用 89 份红、127 份

蓝、没有绿。这个颜色可以写做 rgb(89, 0, 127)，如图 7.4.7 所示。



图 7.4.6　CSS 中最常见的定义颜色的方式是以十六进制数字的形式指定颜色包含的红、绿、蓝的量（另见彩插）

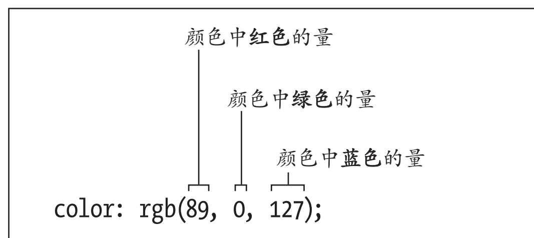


图 7.4.7　表示 CSS 颜色的另一种方式是用 0 ~ 255 之间的数字指示 RGB 数值。首先定义红色，然后是绿色，最后是蓝色

此外，每个值也可以使用百分数表示，不过这种做法很少用到，可能因为 Photoshop 等图像编辑器通常用数字表示 RGB 值。如果你想使用百分数，可以将上面的颜色写作 rgb(35%, 0%, 50%)，因为 89 是 255 的 35%，127 是 255 的 50%。

- 十六进制

我把最常用的方法放在最后讲解，参见图 7.4.8。将这些数字值转化为十六进制，然后将它们合并到一起，再在前面加一个 #，就像 #59007F 这样。（十六进制的 59、00、7F 分别等于 89、0、127。）7F 和 7f 都是允许的写法（我倾向于使用后一种写法，不过很多开发人员和设计人员都选择前一种）。

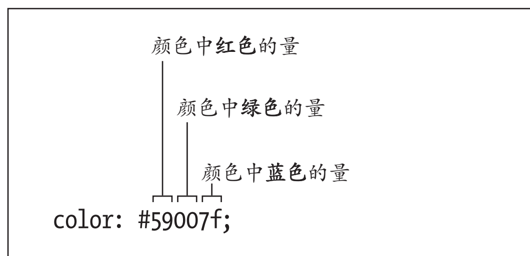


图 7.4.8　CSS 中最常用的定义颜色的方式是用十六进制数指定颜色所包含的红、绿、蓝的量

如果一个十六进制颜色是由三对重复的数字组成的，如 #ff3344，可以缩写为 #f34。这种做法也是一种最佳实践，因为没有理由让代码无谓地变长。

如果十六进制数让你感到头疼，其实不用担心。类似于 Photoshop 这样的工具在选择颜色时可以显示颜色的 RGB 值，以及对应的十六进制数。

7. CSS3 提供的更多指定颜色的方式：RGBA、HSLA 和 HSL

CSS3 引入了另一种指定颜色的方式——HSL，以及通过 RGBA 和 HSLA 设置 alpha 透明度的能力。（使用十六进制记法无法指

示 alpha 透明度。）

• RGBA

RGBA 在 RGB 的基础上加了一个代表 alpha 透明度（alpha transparency）的 A。可以在红、绿、蓝数值后面加上一个用以指定透明度的 0 到 1 之间的小数。语法如下：

```
property: rgba(red, green, blue,
→ alpha transparency);
```

alpha 设置越接近 0，颜色就越透明。如果设为 0，就是完全透明的，就像没有设置任何颜色。类似地，1 表示完全不透明。例如：

```
/* 不透明，和rgb(89, 0, 127);效果相同 */
background: rgba(89,0,127,1);

/* 完全透明 */
background: rgba(89,0,127,0);

/* 25%透明 */
background: rgba(89,0,127,0.75);

/* 60%透明 */
background: rgba(89,0,127,0.4);
```

当然，为了让这些记法正常运行，需要将它们放入一个或多个规则中（参见图 7.4.9）。将 alpha 透明应用到元素的背景颜色上的做法很常见，因为 alpha 透明可以让元素下面的任何东西（如图像、其他颜色、文本等）透过来并混合在一起（参见图 7.4.10）。也可以对其他基于颜色的属性使用 alpha 透明，如 color、border、border-color、box-shadow、text-shadow 等，但需要说明的是，浏览器对它们的支持程度并不相同（现代浏览器的支持情况较好）。

```
/* 设置重复页面背景图像和默认文字颜色 */
body {
    background: url(../img/blueflax.jpg);
    color: #ff0;
}

/* 25% 透明 */
h1 {
    background: rgba(89,0,127,0.75);
}

/* 60% 透明 */
h2 {
    background: rgba(89,0,127,0.4);
}

/* 纯色背景（不透明） */
h3 {
    background: rgb(89,0,127);
}
```

图 7.4.9　这份简单的样式表对整个页面应用了一个重复显示的背景图，设置了默认文字颜色，并对 h1 ~ h3 标题添加了不同的背景。现代浏览器显示的结果如图 7.4.10 所示。正如你即将了解到的，IE9 之前的 Internet Explorer 版本不支持 RGBA，因此它们会忽略对 h1 和 h2 的声明

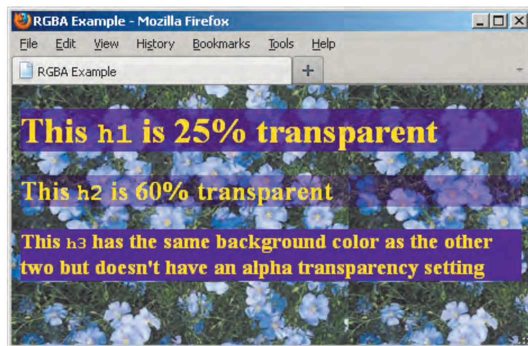


图 7.4.10　这个示例不够真实但很有效，从中我们可以看到页面背景图透过了前两个标题的背景，但没有透过最后一个标题的背景。这三个标题的背景色都是一样的，但由于 alpha 透明度不同，它们看起来就像是深浅不同的三种紫色。（文字是黄色的，这是由为 body 元素设置的 color 属性传递下来的，除非为另一个元素设置的样式规则覆盖它，否则页面上所有的文字都会显示为黄色。）（另见彩插）

7

如你所见，尽管 RGB 颜色值是一样的，但由于设置了不同程度的透明度，颜色在浏览器中显示的样子就不一样了，如图 7.4.10 所示。

HSL 和 HSLA

HSL 和 HSLA 也是 CSS3 中新增的。HSLA 是除 RGBA 以外为颜色设置 alpha 透明度的另一种方式。用 HSLA 指定 alpha 透明度的方法与 RGBA 是一致的。待会儿再看 HSLA，先看看 HSL 是怎样运行的。HSL 代表色相（hue）、饱和度（saturation）和亮度（lightness），其中色相的取值范围为 0 ~ 360，饱和度和亮度的取值均为百分数，范围为 0 ~ 100%，如图 7.4.11 所示。在 CSS 中，语法为：

```
property: hsl(hue, saturation, lightness);
```

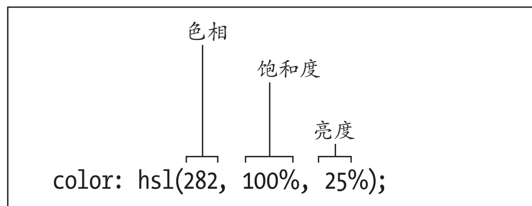


图 7.4.11 对 HSL 格式的分解

你可能猜到了，HSLA 的格式为：

```
property: hsla(hue, saturation,
→lightness, alpha transparency);
```

例如，下面用 HSLA 和 HSL 表示与图 7.4.9 中 RGBA 和 RGB 的例子相同的紫色：

```
/* 25% 透明 */
h1 {
    background: hsla(282,100%,25%,0.75);
}

/* 60% 透明 */
h2 {
    background: hsla(282,100%,25%,0.4);
}

/* 纯色背景（不透明） */
h3 {
    background: hsl(282,100%,25%);
}
```

在现代浏览器中的效果与图 7.4.10 所示的一样。

可以将色相值想成一个圆环中的度数，0 和 360 在顶端相遇。这意味着 0 和 360 均表示同一种颜色（红色）。（不要将 HSL 与 HSB（又称 HSV）[①]弄混，它们是相似的，但并不相同。）

如何构想 HSL

学习 HSL 的逻辑需要花些时间，但是一旦找到感觉，你会发现它比其他的格式更容易使用。Brandon Mathis 在其 HSL Color Picker 网站（http://hslpicker.com）的“为何使用 HSL”部分对这种格式有一段精彩的解释。他写道：

> “选择一个 0 到 360 之间的色相，并将饱和度设为 100，亮度设为 50，就会得到这种颜色最纯的形式。降低饱和度，颜色就会向灰色变化。增加亮度，颜色就会向白色变化；减少亮度，颜色就会向黑色变化。”

随着你在圆环上移动，可以得到一些不同的颜色，例如：

- 红色为 hsl(**0**,100%,50%);
- 黄色为 hsl(**60**,100%,50%);
- 绿色为 hsl(**120**,100%,50%);
- 青色为 hsl(**180**,100%,50%);
- 蓝色为 hsl(**240**,100%,50%);
- 紫红色为 hsl(**300**,100%,50%);

① HSB 中的 B 表示明度（brightness），HSV 中的 V 表示亮度（value）。——译者注

并非所有的图像编辑器都可以在对话框中指定 HSL（可以为 Photoshop 安装一个插件）。不过，通过 Mathis 强大的免费在线工具 HSL Color Picker，可以选取颜色，获取其 HSL 值、十六进制数值和 RGB 值，还可以输入这些格式的颜色值，查看颜色的变化。另一种类似的工具位于 www.workwithcolor.com/hsl-color-picker-01.htm。它显示了一个颜色圆环，帮你更好地理解 HSL。（相较而言，HSL Color Picker 将颜色显示在一条直线上。）通过在线搜索还可以找到更多其他的颜色工具。

Internet Explorer 中的 RGBA、HSL 和 HSLA

不幸的是，如同标准世界那些最新的发展常常遭遇的情形，IE9 之前的 Internet Explorer 版本均不支持上述功能。它们无法理解这些记法，因此会忽略这些声明。

对于 IE9 之前的版本，通过一些变通的方法可以使用 RGBA 和 HSLA。但 HSL 还是用不了，因此只能继续使用十六进制数（或 RGB）指定颜色。在 IE9 之前的版本中使用 RGBA 和 HSLA 有三种选择（但一次只能选择一种）。

- 什么都不做，让页面在这些版本中看起来完全不一样。
- 为它们提供后备的颜色声明，即显示纯色，而不是透明的颜色，参见图 7.4.12。
- 通过为它们指定一些专门的声明模拟 alpha 透明。这些声明多为 IE 特有的 CSS。现代浏览器仍将使用标准的 CSS，参见图 7.4.13。

```
/* background 声明的顺序是很重要的。IE 的旧版
   本使用第一条声明，而现代浏览器对这两条声
   明都理解，但仅应用第二条声明（因为它在最
   后）。 */

h1 {
    background: #59007f;
    background: rgba(89,0,127,0.75);
}
```

图 7.4.12　现代浏览器以 RGBA 声明呈现颜色，因为这条声明出现在默认的十六进制背景设置的后面（现代浏览器也理解十六进制的设置，因此顺序就很重要了）。但 IE9 之前的 IE 版本不理解这种记法，它们会忽略 RGBA 设置，因此十六进制背景颜色会生效。也可以使用 RGB（并非 RGBA）替代第一行里的十六进制数，但正如上文提到过的，十六进制数是最常用的指定不透明颜色的方式

```
/* 如果你像我一样，早就想哭了。除了第二条声明
→以外，其他所有的声明都是为 IE 的旧版本模拟
→ alpha 透明的。 */

h1 {
    background: transparent;
    background: rgba(89,0,127,0.75);

    /* IE8 */
    -ms-filter:
    "progid:DXImageTransform.
    →Microsoft.gradient(startColorstr=
    →#BF59007F,endColorstr=#BF59007F)";

    /* IE6 和 7 */
    filter: progid:DXImageTransform.
    →Microsoft.gradient(startColorstr=
    →#BF59007F,endColorstr=#BF59007F);

    zoom: 1;
}
```

图 7.4.13　不要崩溃！这段为 IE9 之前版本准备的冗长的代码“三明治”将标准 RGBA 记法（未突出显示）包围起来。旧版本的 IE 照例会忽略它们不理解的声明。类似地，现代浏览器会忽略它们不理解的 -ms-filter、filter 和 zoom 值。要确保这段代码正常工作，声明的顺序很重要

最后一种选择结合使用了 Internet Explorer 的 Gradient 滤镜和一些不为其他浏览器所理解的专有代码。这意味着现代浏览器将忽略这些声明，还是使用标准的记法（在本例中为 `background: rgba(89,0,127,0.75);`，它覆盖了之前的 `background` 值）。注意这些声明应以所示的顺序排列，从而使它们在现代浏览器和旧的浏览器中都能正常工作（参见图 7.4.13）。

我不会费力去解释 IE Gradient 滤镜的语法，因为它们极其令人费解，你几乎不可能手写这些代码。我也不会这样做。不过，一些免费的在线工具可以帮到你（随着你深入学习 CSS3，你将看到更多这样的工具）。

Michael Bester 的 RGBa & HSLa CSS Generator for Internet Explorer（http://kimili.com/journal/rgba-hsla-css-generator-for-internet-explorer）就是这样一款工具。据其解释，你输入 RGBA 或 HSLA 声明后，该工具就能创建等价的为 IE9 之前的版本准备的 CSS。然后，将它们复制到你的样式表中即可。注意：该工具创建的代码不包含为现代浏览器准备的标准 RGBA 或 HSLA 声明。因此，你需要自己将它直接添加到 `background: transparent;` 的后面，如图 7.4.13 所示。同时，如 Michael 指出的，也可以将为 IE9 之前的版本准备的 CSS 放到单独的样式表中，再通过所谓的条件注释（conditional comment）将它们加载进来（详细信息参见 http://reference.sitepoint.com/css/conditionalcomments）。

提 示 Internet Explorer 滤镜（如图 7.4.13 所示的 Gradient 滤镜）会影响浏览器的性能，因为它们需要额外的处理能力。如果页面中只有少量的元素应用了滤镜，也不会产生明显的问题，但如果滤镜的数量较多，就可能产生延时。是否产生延时也取决于页面内的其他元素。因此，构建页面的时候要注意这一点，如果你在 IE 中看到页面明显变慢，可以关掉滤镜看看是不是由它引起的问题。IE 滤镜有时还会产生其他一些意想不到的副作用，如对文本呈现的质量产生不良影响。需要说明的是，这些滤镜不会影响到其他的浏览器，因为其他浏览器并不理解它们。

第 8 章

操作样式表文件

本章内容

- 创建外部样式表
- 链接到外部样式表
- 创建嵌入式样式表
- 应用内联样式
- 位置的重要性
- 使用与媒体相关的样式表
- 提供替代的样式表
- 借鉴其他人的灵感：CSS

在开始定义样式表之前，要知道如何创建和使用包含这些样式的文件。在本章中，你将学习如何创建样式表文件，如何将 CSS 应用到多个网页（包括整个网站）、单个页面或单独的 HTML 元素。这三种应用分别通过三种方法实现：外部样式表（首选方法）、嵌入样式表和内联样式（最不可取的方法）。

在随后的几章，你将学习如何创建样式表中的内容。

8.1 创建外部样式表

外部样式表非常适合给网站上的大多数页面或者所有页面设置一致的外观。可以在一个外部样式表中定义全部样式，然后让网站上的每个页面加载这个外部样式表，从而确保每个页面都有相同的设置。尽管后面你将学到嵌入样式表和内联样式表，但从外部样式表为页面添加样式才是最佳实践，推荐你使用这种方法（允许偶尔的例外）。

创建外部样式表

(1) 在文本编辑器中创建一个新文档，如图 8.1.1 所示。

(2) 按照第 7 章的讲解，为网页定义样式规则。同时，根据需要在 CSS 中添加注释（参见图 8.1.1）。

(3) 将文档以纯文本格式保存在希望放置的目录中。尽管任何名称都是允许的，但应该为该文件添加 .css 的扩展名，表明这是一个层叠样式表，如图 8.1.2 所示。

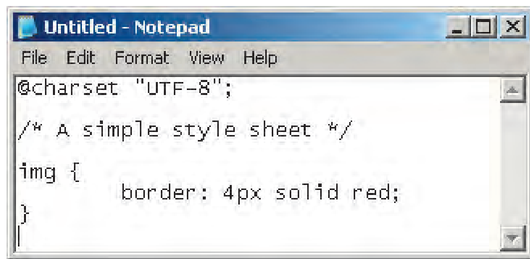


图 8.1.1 使用任何你喜欢的文本编辑器创建样式表。这是记事本（一种旧版本）。大多数人在创建 HTML 和 CSS 文档时使用相同的编辑器。/* */ 之间的文本是 CSS 注释，它既不会影响页面的显示，也不会出现在页面中

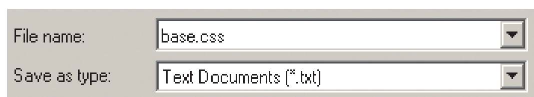


图 8.1.2　一定要将 CSS 文件保存为以 .css 为扩展名的纯文本格式文件（可能称为 Text Document、Plain Text、ASCII 等）

提 示　可以以任何名称为样式表文件命名。base.css 和 global.css 是两种常见的样式表名称，它们通常包含应用于网站大多数页面的样式规则。网站制作者通常创建一些为某些区块所特有的附加 CSS 文件，作为对基本样式的补充。例如，对于一个商业网站，products.css 包含的可能是为产品相关页面准备的样式规则。无论选择什么文件名，一定不要包含空格。

提 示　外部样式表要么是通过链接引用的（如 8.2 节中介绍的），要么是导入的（通过 @import），不过不推荐你导入它们。@import 指令会影响页面的下载速度和呈现速度，在 Internet Explorer 中影响更为明显（参见 Steve Souders 对此问题的讨论：www.stevesouders.com/blog/2009/04/09/dont-use-import/）。

提 示　样式表开头处的 @charset 并不总是必需的，不过，总是在样式表中包含它也没有任何坏处，参见图 8.1.1。不过，如果样式表中包含非 ASCII 字符（使用 CSS 生成内容（一项高级主题）或名称中含有特殊字符的 Web 字体时就会遇到这种情况），就**必须**包含它。出于这种原因，你可以选择总是包含 @charset，以免后来样式表需要它时再回过头来添加，此外，一定要将它放在样式表的第一行。不过，不要在嵌入样式表或内联样式（本章后文会讲到）中包含 @charset。

8.2　链接到外部样式表

创建了样式表（参见图 8.2.1）之后，需要将它加载到 HTML 页面中去，从而为内容应用这些样式规则。要做到这一点，最好的方式是链接到样式表（参见图 8.2.2）。

```
@charset "UTF-8";

img {
    border: 4px solid red;
}
```

图 8.2.1　这是本章前面创建的外部样式表 base.css（去掉了注释“A simple style sheet”，这对 HTML 的显示不会产生任何影响）。不必担心不理解这里的属性和值（它们表示“为所有的 img 元素创建红色的实线边框”）。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>El Palau de la Música</title>
    <link rel="stylesheet"
    → href="base.css" />
</head>
<body>
<article>
    <h1>El Palau de la Música</h1>

    <img src="img/palau250.jpg" width="250"
    → height="163" alt="El Palau de la
    → Música" />
    <img src="img/tickets.jpg" width="87"
    → height="163" alt="The Ticket Window" />

    <p>I love the <span lang="es">Palau de la
    → Música</span>. It is ornate and gaudy
    → and everything that was wonderful
    → about modernism. It's also the home
    → of the <span lang="es">Orfeó Català
    → </span>, where I learned the benefits
    → of Moscatell.</p>
</article>
</body>
</html>
```

图 8.2.2　link 元素位于 HTML 文档的 head 部分。页面可以包含一个以上的 link 元素，但使用它的次数最好尽可能地少，让页面得以更快地加载

链接到外部样式表的步骤

(1) 在每个希望使用样式表的 HTML 页面的 head 部分，输入 `<link rel="stylesheet"`。

(2) 输入一个空格，然后输入 href="*url.css*"，其中 *url.css* 是 CSS 样式表的名称（参见上一节）。

(3) 输入一个空格和 />。（如果你愿意，也可以不输入空格，直接输入 >；这两种形式都是 HTML5 所允许的，它们的效果也完全一样。）

提示 对外部样式表进行修改时，所有引用它的页面也会自动更新（参见图 8.2.3 和图 8.2.4）。这是外部样式表的神奇力量！

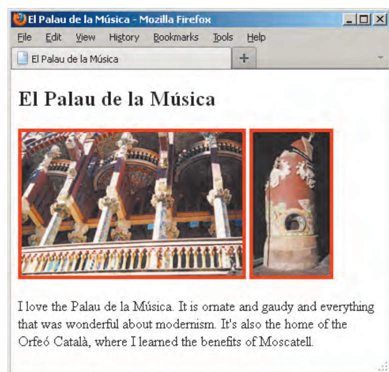


图 8.2.3 将这个样式规则（4 像素宽的红色实线边框）应用于每个 img 元素（另见彩插）

提示 外部样式表的另一个好处是，一旦浏览器在某个页面加载了它，在随后浏览引用它的页面时，通常无需再向 Web 服务器请求该文件。浏览器会将它保存到缓存里，也就是保存到用户的计算机里，并使用这个版本的文件。这样做可以加快对页面的加载。不过，不必担心。如果随后对样式表作了修改，再将它传到 Web 服务器，浏览器就会下载更新后的文件，而不是使用缓存的文件（从技术上讲也有例外，但通常不会遇到这种情况）。

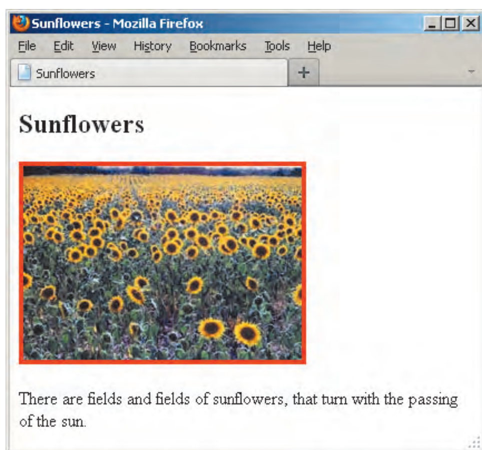


图 8.2.4 其他文档也可以链接到同一个样式表，并拥有同样的样式（另见彩插）

提示 出于简化的目的，这个例子中的链接假定 HTML 页面与 base.css 位于同一个目录（参见图 8.2.2）。不过，实践中最好将样式表组织在子文件夹里，而不是与 HTML 页面混在一起。常见的样式表文件夹名称包括 css、styles 等，你也可以按照自己的意愿对其进行命名，只要你认为该名称放在链接的 href 值里是合适的。例如，如果 base.css 位于名为 css 的文件夹里，而 HTML 位于该文件夹的上一级，那么 link 元素就应该写做 `<link rel="stylesheet" href="`**`css`**`/base.css" />`。

提示 外部样式表里的 URL 是相对于服务器上该样式表文件的位置的，而不是相对于 HTML 页面的位置的。你将在第 10 章学习 CSS 背景图像时看到有关的实例。

提示 外部样式表中的规则可能被 HTML 文档内的样式覆盖。8.5 节中总结了以不同方式应用的样式的相对影响力。

提示　可以同时链接到多个样式表。如果不同的文件中有相互冲突的显示规则，则靠后的文件中的规则具有更高的优先级。

提示　可以为链接的样式表提供替代版本，让访问者从中选择。参见 8.7 节。

提示　可以通过设置 media 属性将样式表限制在特定类型的输出上。详细信息参见 8.6 节。

提示　HTML 早期版本要求在 link 元素定义中包含 type="text/css"，但 HTML5 不要求这样做，因此可以像代码示例那样忽略它。

8.3　创建嵌入样式表

嵌入样式表是在页面中应用 CSS 的第二种方式。它允许在 HTML 文档里直接设置样式（通常位于 head 部分），如图 8.3.1 所示。由于这些样式只在一个网页里存在，因此不会像外部样式表中的规则那样应用到其他的页面，同时，缓存的好处也不存在了。如上文所述，对于大多数情况，外部样式表是推荐的方式，但理解其他的选择以备不时之需也是很重要的。

创建嵌入样式表的步骤

(1) 在 HTML 文档的 head 部分输入 <style>。

(2) 根据需要，定义任意数量的样式规则（参见 7.1 节）。

(3) 输入 </style> 结束嵌入样式表（参见图 8.3.1）。

提示　当且仅当 style 元素出现在 link 元素后面的时候，嵌入样式表中的样式才会覆盖外部样式表中的样式。详细信息参见 8.5 节。

```
...
<head>
    <meta charset="UTF-8" />
    <title>El Palau de la Música</title>
    <style>
    img {
       border: 4px solid red;
    }
    </style>
</head>
<body>
...
    <img src="img/palau250.jpg" width="250"
    → height="163" alt="El Palau de la
    → Música" />
    <img src="img/tickets.jpg" width="87"
    → height="163" alt="The Ticket Window" />
...
</body>
</html>
```

图 8.3.1　使用嵌入样式表时，style 元素及其包围的样式规则通常位于文档的 head 部分。浏览器对页面的呈现方式与使用外部样式表时是一样的，参见图 8.3.2。注意，嵌入样式表不应在其开头处（或其他任何位置）包含 @charset 声明

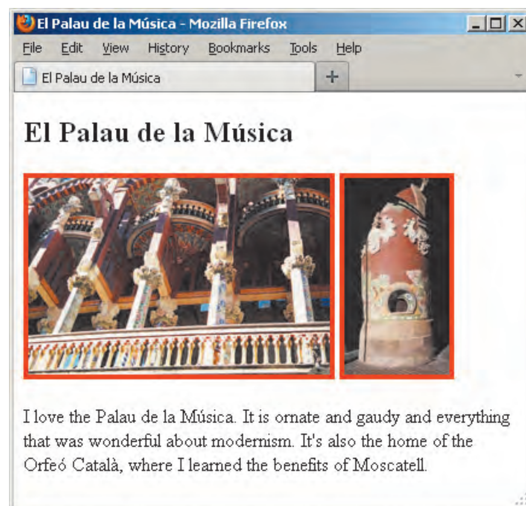


图 8.3.2　结果与链接到外部样式表中的样式时完全一样。区别在于，其他网页无法利用这个页面中定义的样式

提示 嵌入样式表是为页面添加 CSS 的次选方式。（也有例外的情况，如特定条件下拥有极大流量的网站。）推荐的方法是加载外部样式表。更多信息参见 8.2 节。

提示 还可以直接对单个 HTML 元素应用样式，但强烈建议不要这样做。详细信息参见 8.4 节。

提示 从技术上说，在页面的 body 部分嵌入样式表也是可行的，但应尽可能避免这种做法。将内容（HTML）、表现（CSS）和行为（JavaScript）分离是一种最佳实践，而将 HTML 和 CSS 混在一起就会打破这种原则。从实际情况来看，在外部样式表中维护 CSS 比在嵌入样式表中维护 CSS 更为容易。

8.4 应用内联样式

内联样式是在 HTML 中应用 CSS 的第三种方式。不过，应当最后考虑这种方式，因为它将内容（HTML）和表现（CSS）混在了一起，严重地违背了最佳实践，如图 8.4.1 所示。内联样式只影响一个元素，如图 8.4.2 所示，因此使用它将失去外部样式表的重要好处：一次编写，到处可见。设想要对大量 HTML 做简单的文字颜色的改变，就需要对这些页面逐一进行检查和修改，可见内联样式不被经常使用的原因。

不过，如果你想快速地测试某种样式，以便随后将它从 HTML 中搬到更易于长期维护的外部样式表中（假定你对结果满意），内联样式就能派上用场了。

```
...
<head>
    <meta charset="UTF-8" />
    <title>El Palau de la Música</title>
</head>
<body>
...
    <img src="palau250.jpg" width="250"
    → height="163" alt="El Palau de la
    → Música" style="border: 4px solid
    → red" />
    <img src="tickets.jpg" width="87"
    → height="163" alt="The Ticket Window" />
...
</body>
</html>
```

图 8.4.1 内联样式规则只影响单个元素（在本例中为第一个 img 元素）

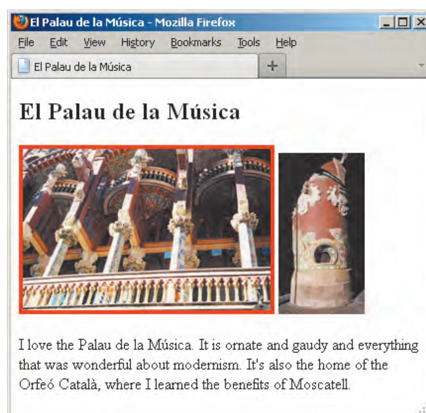


图 8.4.2 只有第一个图像拥有边框。如果要让其他元素也有这种效果，就要在每个 img 元素中单独加上 style="border: 4px solid red"。如你所见，内联样式的效率很低，对全站的内联样式统一进行更新是一件很头疼的事

应用内联样式的步骤

(1) 在希望进行格式化的 HTML 元素中，输入 style="。

(2) 创建一个样式规则，但不要包含花括号和选择器。不需要选择器是因为直接将样式放入目标元素中了。

8

(3) 要创建其他样式定义，输入 ;（分号）并重复第 (2) 步。

(4) 输入后引号 "。

提示　注意，不要将等号与冒号弄混。由于它们都表示赋值，因而很容易不小心用错。

提示　不要忘了用分号分隔多条属性定义。

提示　不要忘了用引号包围样式定义。

提示　内联样式的优先级高于其他所有样式，除非其他地方与之冲突的样式标记了 !important（参见 8.5 节）。

提示　如果要在内联样式声明中指定字体，对于有多个单词的字体名称，要用单引号包围，以避免与 style 元素的双引号发生冲突。不能在这两个地方使用相同类型的引号。

提示　或许内联样式最为常见的使用场景是在 JavaScript 函数中为元素应用内联样式，从而为页面某个部分添加动态行为。可以通过 Firebug 或 Chrome 的开发者工具查看这些生成的内联样式。在大多数情况下，应用这些样式的 JavaScript 同 HTML 是分离的，因而仍然保持了内容（HTML）、表现（CSS）和行为（JavaScript）分离的原则。

8.5　位置的重要性

将多个样式规则应用于同一元素的情况并不少见，大型网站更是如此（对于它们，需要更多的精力去维护 CSS）。正如在 7.3 节中提到的，样式的位置可以打破继承和特殊性之间的僵局。基本规则是：在其他条件相同的情况下，越晚出现的样式优先级越高，或者说越重要（参见图 8.5.1 至图 8.5.4）。

因此，内联样式拥有最高的优先级，它将覆盖其他任何位置与之冲突的样式。

在嵌入的 style 元素中，任何 @import 规则都会被同一 style 元素中出现的单独样式规则覆盖（因为根据定义，这些样式规则必须出现在 @import 规则后面）。

```
…
<head>
    <title>El Palau de la Música</title>
    <link rel="stylesheet"
    → href="base.css" />
    <style>
    img {
        border-style: dashed;
    }
    </style>
</head>
…
```

图 8.5.1　在这个例子中，style 元素出现得最晚。因此，它的规则的优先级比 base.css 样式表中的高（在有冲突的规则的继承性和特殊性相同的情况下）

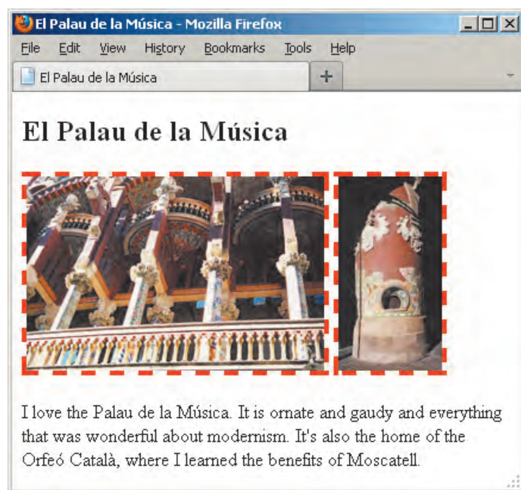


图 8.5.2　style 元素中定义的虚线边框优先于链接的 base.css 中定义的实线边框

```
...
<head>
    <title>El Palau de la Música</title>
    <style>
    img {
       border-style: dashed;
    }
    </style>
    <link rel="stylesheet" href="base.
css" />
</head>
...
```

图 8.5.3 这里，链接的样式表最后出现，其优先级高于 style 元素中的规则（在其他情况都相同的时候）

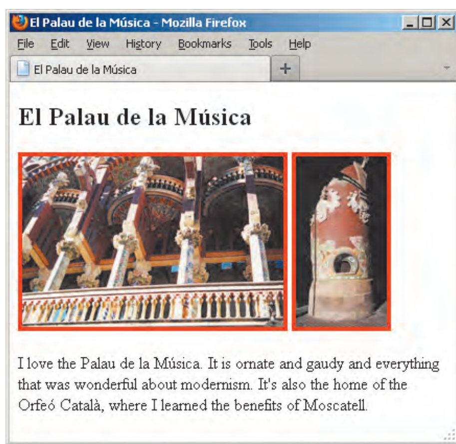


图 8.5.4 base.css 样式表中定义的实线边框优先于内部 style 元素中定义的虚线边框

嵌入的 style 元素与任何链接的外部样式表之间的关系取决于它们的相对位置。如果 link 元素在 HTML 代码中出现得晚，它就会覆盖 style 元素。如果 link 元素出现得早，style 元素（及其包含的任何导入样式表）就会覆盖链接的样式表中的规则。

外部样式表也可以包含 @import 规则（不过，上文提到，出于性能上的考虑，不推荐这种用法）。在这种情况下，导入的规则会被外部样式表中的其他规则覆盖（因为根据定义，这些规则必须出现在 @import 规则的后面）。它们与文档中其他样式表的关系照例由外部样式表链接的位置决定。

关于相互冲突的样式的顺序对优先级的影响，有一种例外情况，就是标记 !important 的样式总是具有最高的优先级，无论它出现在最前、最后，还是中间。例如，p { margin-top: 1em !important; }。不过，要尽量避免这种用法。它让声明变得太强，如果要覆盖这样的样式，就不得不借助于更长的规则。

唯一一处值得在声明中使用 !important 的地方是用户样式表。是的，网站的访问者可以为其浏览器创建他们自己的样式表，且这种样式的优先级很高。例如，访问者可能想使用特定大小的字号，或调整文本颜色与背景颜色之间的对比度。不过，大多数用户并不知道他们可以这样做，因此这种情况也是很不常见的。

8.6 使用与媒体相关的样式表

可以指定一个只用于特定输出的样式表，如只用于打印，或只用于用浏览器在屏幕上查看。例如，可以创建一个具有打印和屏幕版本共有特征的通用样式表，再创建单独的打印样式表和屏幕样式表，分别包含只用于打印的属性和只用于屏幕显示的属性。

指定与媒体相关的样式表的步骤

(1) 在 link 或 style 开始标记中添加 media="*output*"，其中 *output* 可以是 print、screen 或 all（尽管还有其他一些选项，但这些是最常见的），参见图 8.6.1。多个值之间用逗号分隔。

(2) 也可以在样式表中使用 @media 规则，参见图 8.6.2。这种方法不需要在 link 元素中指定媒体类型。

```
...
<head>
    <meta charset="UTF-8" />
    <title>El Palau de la Música</title>
    <link rel="stylesheet" href="base.css"
    ➝ media="screen" />
    <link rel="stylesheet" href="print.css"
    ➝ media="print" />
</head>
<body>
...
    <img src="img/palau250.jpg" width="250"
    ➝ height="163" alt="El Palau de la
    ➝ Música" />
    <img src="img/tickets.jpg" width="87"
    ➝ height="163" alt="The Ticket Window" />
...
</body>
</html>
```

图 8.6.1　通过对 link 元素添加 media 属性，可以将样式表限制于某种特定的输出。在这个例子中，使用浏览器查看页面时，base.css 会起作用（由于使用了 media="screen"）；打印页面时，print.css 会起作用（由于使用了 media="print"）

提 示　media 属性的默认值是 all，因此声明 media="all" 是多余的。换句话说，除非需要让样式表具有特殊的输出，否则可以将 media 属性留空。有的编码人员倾向于始终显式地包含 media="all"。

提 示　有 9 种可能的输出类型：all、aural、braille、handheld、print、projection、screen、tty 和 tv。浏览器对它们的支持程度各不相同（大多只有少量的支持）。实际上，用得上的只有 screen 和 print（或许还应加上 all），浏览器对它们的支持情况都很好。另一方面（可以这么说），设备对 handheld 的支持程度并不高，因此在为移动设备设计时，通常使用 screen 而不是 handheld（参见第 12 章）。projection 类型是为投影仪和其他类似的设备准备的，Opera 的投影模式 Opera Show 支持这种类型。

```
@charset "UTF-8";

/* 针对所有媒体的样式 */
img {
    border: 4px solid red;
}

p {
    color: green;
    font-style: italic;
}

/* 打印样式表 */
@media print {
    img {
       display: none;
    }

    p {
       color: black;
    }
}
```

图 8.6.2　样式表中的 @media 规则是指定其他媒体类型的另一种方式（参见第 12 章）。在这个例子中，上面是影响所有媒体类型（包括打印）的样式，下面是专门用于打印的样式。在页面的打印预览模式将看到，图片并未显示（display: none 将它们隐藏了），段落文字是黑色的斜体文字。font-style: italic 声明也应用于打印模式，因为打印样式表并未指定其他的 font-style

提 示　更多关于创建打印样式表的信息，参见 Christian Krammer 的文章：www.smashing-magazine.com/2011/11/24/how-to-set-up-a-print-style-sheet/。

8.7　提供替代的样式表

可以链接到多个样式表（如图 8.7.1 所示），然后让访问者选择他们最喜欢的一个。规范允许设置一组基本的持久性样式（persistent style），如图 8.7.2 所示（无论访问者的首选项是什么，都会应用它们）；规范允许设置一组默认（或称首选）样式，如图 8.7.3 所示，如果访问者没有进行选择，就会应用它

们；此外，规范还允许设置一个或多个替代（alternate）样式表，如图 8.7.4 所示。访问者可以选择这些替代样式表（参见图 8.7.5），从而使首选样式失效（但不影响持久性样式）。替代样式表可以让网站拥有不同的主题。

```
...
<head>
    <meta charset="UTF-8" />
    <title>Palau de la Música</title>
    <link rel="stylesheet"
    → href="base.css" />
    <link rel="stylesheet" href=
    → "preferred.css" title="Dashed" />
    <link rel="alternate stylesheet"
    → href="alternate.css"
    → title="Dotted" />
</head>
...
```

图 8.7.1　按顺序定义了基本的（或称持久性的）样式表、首选的（或称自动的）样式表和替代样式表。每个样式表都需要单独的 link 元素

```
img {
    border: 4px solid red;
}
```

图 8.7.2　这个 CSS 文件（base.css）将作为持久性样式表的例子，无论访问者做什么，都会应用它

```
img {
    border-style: dashed;
}
```

图 8.7.3　当访问者转入这个页面时，在加载完 base.css 后将默认加载这个样式表（preferred.css）

```
img {
    border-style: dotted;
}
```

图 8.7.4　如果访问者愿意，它们可以加载这个样式表。文件名为 alternate.css，尽管这是一种通行做法，但是也可以为它起你喜欢的名称

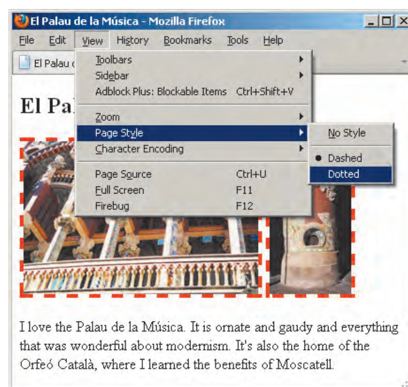


图 8.7.5　页面加载以后，图像有虚线边框（首选样式的值覆盖了基本样式的实线值，但基本样式中的颜色被保留下来了）。如果用户选择 Dotted，就将使用替代样式表（另见彩插）

提供替代样式表的步骤

(1) 要指定基本样式表（不论访问者的首选项是什么），就使用 8.2 节中描述的简单语法，不带 title 属性。

(2) 要指定可以被其他样式表替代的首选样式表，就在 link 元素中添加 title="*label*"，其中 *label* 是标识首选样式表的名称。

(3) 要指定替代样式表，就在 link 元素中使用 rel="alternate stylesheet" title="*label*"，其中 *label* 是标识替代样式表的名称。

提示　不必为了提供替代样式表而提供首选样式表。这个例子也可以只有指向 base.css 和 alternate.css 的 link 元素。类似地，也可以不指定持久性样式表，只指向 preferred.css 和 alternate.css。也可以有一个以上的替代样式表。

提示　Firefox（参见图 8.7.5）和 Opera 提供了在样式表之间进行切换的简单方法。对于其他浏览器，就只能使用 JavaScript 解决方案了。可以在网上搜索“style sheet switcher”（样式表切换器），寻找可供使用的代码。

提 示　几年以前，替代样式表是让用户选择字号的一种方法。现在，不借助替代样式表在浏览器中增大字号已经变得相对容易，而且有越来越多的用户意识到了这类功能（主要为页面缩放）。

提 示　还可以加载用以打印网页的样式表。详细信息参见 8.6 节。

8.8　借鉴他人灵感：CSS

在第 2 章中，你学到了如何查看网页的源代码。查看其他人的 CSS 也不太困难。

查看其他设计人员的 CSS 代码

(1) 首先查看页面的 HTML 代码（如图 8.8.1 所示）。关于查看 HTML 源代码的更多信息，参见 2.8 节。

如果 CSS 代码是嵌入样式表，就已经能看到它们了。

(2) 如果 CSS 是外部样式表，就在 HTML 中找到对它的引用，再点击其文件名（参见图 8.8.1）。样式表就显示在浏览器窗口中了（如图 8.8.2 所示）。如果愿意，可以复制这些代码，再粘贴到文本编辑器中。



图 8.8.1　查看包含你感兴趣的样式表的 HTML 页面源代码，再点击样式表的文件名

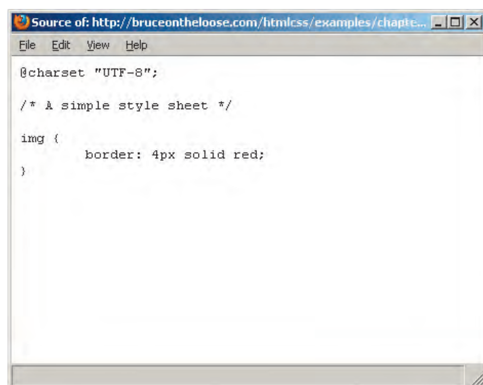


图 8.8.2　样式表显示在浏览器窗口中

提 示　同 HTML 一样，可以从其他设计人员的代码中借鉴灵感，然后编写自己的样式表。不过，查看其他人的代码时也要留心。这些代码放在万维网上并不意味着它们就是编写某一效果的代码的最佳方式（尽管其作者希望如此）。

提 示　现代浏览器允许像图中显示的那样直接在 HTML 代码中点击样式表的名称。要在旧的浏览器中查看样式表，可能需要复制 link 元素中显示的 URL，再粘贴到浏览器的地址栏（替换 HTML 文件名），然后按回车键。如果样式表的 URL 是相对地址（参见 1.7 节），就需要将网页的 URL 和样式表的相对 URL 结合在一起，形成样式表的完整 URL。

提 示　使用现代浏览器提供的开发者工具，可以快速查看页面的 CSS。大多数浏览器都捆绑了这类工具。Firefox 中有一个实现此功能的扩展，名为 Firebug（参见第 20 章）。

第 9 章 定义选择器

9

本章内容

- ❑ 构造选择器
- ❑ 按名称选择元素
- ❑ 按类或 ID 选择元素
- ❑ 按上下文选择元素
- ❑ 选择元素的一部分
- ❑ 按状态选择链接元素
- ❑ 按属性选择元素
- ❑ 指定元素组
- ❑ 组合使用选择器
- ❑ 选择器回顾

正如 7.1 节中看到的，CSS 样式规则有两个主要部分。选择器决定格式化将应用于哪些元素，而声明则定义要应用的样式化。在本章中，你将学习如何定义 CSS 选择器。

最简单的选择器可以对给定类型的所有元素（如所有的 `h1` 标题）进行样式化，更复杂的选择器可以根据元素的类或 ID、上下文、状态等来应用格式化规则。

在定义了选择器以后，可以根据第 10 章 ~ 第 14 章的讲解创建声明（包括实际的属性和值）。本书的其余部分会讨论一些比较特殊的样式属性。在此之前，我们将在例子中使用非常简单且意义明确的 `{color: red;}`。

9.1 构造选择器

选择器决定样式规则应用于哪些元素。例如，如果要对所有的 `p` 元素添加 Times 字体、12 像素高的格式，就需要创建一个只识别 `p` 元素而不影响代码中其他元素的选择器。如果要对每个区域中的第一个 `p` 设置特殊的缩进格式，就需要创建一个稍微复杂一些的选择器，它只识别作为页面中每个区域的第一个元素的 `p` 元素。

选择器可以定义五个不同的标准来选择要进行格式化的元素。

- ❑ 元素的类型或名称，如图 9.1.1 所示。
- ❑ 元素所在的上下文，如图 9.1.2 所示。
- ❑ 元素的类或 ID，如图 9.1.3 和图 9.1.4 所示。
- ❑ 元素的伪类或伪元素，如图 9.1.5 所示（放心，我将在后面解释伪类和伪元素）。
- ❑ 元素是否有某些属性和值，如图 9.1.6 所示。

为了指出目标元素，选择器可以使用这五个标准的任意组合。在大多数情况下，只使用一个或两个标准即可。另外，如果要对几组不同的元素应用相同的样式规则，可以将相同的声明同时应用于几个选择器（参见 9.8 节）。

本章其余部分将详细解释如何定义选择器。

要选择的元素的名称

```
h1 {
   color: red;
}
```

图 9.1.1　最简单的选择器类型就是要格式化的元素的类型的名称（在这个例子中为 h1）

上下文
要选择的元素的名称

```
h1 em {
   color: red;
}
```

图 9.1.2　这个选择器使用上下文。这个样式只应用于 h1 元素中的 em 元素。其他地方的 em 元素不受影响

类

```
.very {
   color: red;
}
```

ID

```
#gaudi {
   color: red;
}
```

图 9.1.3　第一个选择器选择所有属于 very 类的元素。换句话说，就是所有在开始标记中包含 class="very" 的元素。第二个选择器选择 id 为 gaudi 的那个元素，也就是在开始标记中指定 id="gaudi" 的元素。回忆一下，一个 id 在每个页面中只能出现一次，而一个 class 可以出现任意次数

要选择的元素的名称
类

```
em.very {
   color: red;
}
```

要选择的元素的名称
ID

```
article#gaudi {
   color: red;
}
```

图 9.1.4　通过在 class 或 id 选择器前面添加目标元素的名称，可以增强选择器的特殊性。在这个例子中，第一个选择器仅选择属于 very 类的 em 元素，而不是属于 very 类的任何元素。类似地，第二个选择器仅选择 id 为 gaudi 的 article 元素。除非确有必要，一般不要使用这种方法。图 9.1.3 的示例中的特殊性弱一些的选择器是更好的方法

名称
伪类

```
a:link {
   color: red;
}
```

图 9.1.5　在这个例子中，选择器选择属于 link 伪类的 a 元素（即还没有访问过的 a 元素）

名称
属性

```
a[name] {
   color: red;
}
```

图 9.1.6　可以在选择器中使用方括号添加关于目标元素的属性或值的信息

9.2 按名称选择元素

选择要格式化的元素，最常用的标准可能是元素的名称或类型。例如，可能要让所有的 h1 元素以大字号、粗体显示，让所有的 p 元素以无衬线字体显示（如图 9.2.1 所示）。

```
<!DOCTYPE html>
<html lang="en">
<head>
...
</head>
<body>
...
<article class="about">
    <h1>Antoni Gaudí</h1>

    <p>Many tourists are drawn to Barcelona to see Antoni Gaudí's incredible architecture.</p>
    <p>Barcelona <a href="http://www.gaudi2002.bcn.es/english/" rel="external">celebrated the 150th
    ➝ anniversary</a> of Gaudí's birth in 2002.</p>

    <section class="project">
       <h2 lang="es">La Casa Milà</h2>
       <p>Gaudí's work was essentially useful. <span lang="es">La Casa Milà</span> is an apartment
       ➝ building and <em>real people</em> live there.</p>
    </section>

    <section class="project">
       <h2 lang="es">La Sagrada Família</h2>
       <p>The complicatedly named and curiously unfinished Expiatory Temple of the Sacred Family is
       ➝ the <em>most visited</em> building in Barcelona.</p>
    </section>
</article>
...
```

图 9.2.1 这段 HTML 代码包含两个 h2 元素（或许你对 lang 属性感到疑惑。它指明内容使用的是页面默认语言以外的另一种语言。页面默认语言是在每页开头处紧跟在 DOCTYPE 后面的 html 元素中指定的。在这个例子中，每个 h2 上的 lang="es" 表示其内容为西班牙语。）

按照类型选择要格式化的元素的方法

输入 *selector*，其中 *selector* 是目标元素的类型名称（不含任何属性），如图 9.2.2 所示。

```
h2 {
    color: red;
}
```

图 9.2.2 这个选择器会选择文档中所有的 h2 元素，并让它们显示为红色，参见图 9.2.3

提示 除非指定其他情况（使用本章其余部分介绍的技术），指定类型的所有元素都将被格式化，无论它们出现在文档的什么位置。



图 9.2.3 所有的 h2 元素都显示为红色（另见彩插）

提示 并非所有的选择器都需要指定元素的名称。如果要一类元素进行格式化，而不管属于这个类的元素的类型，就可以从选择器中去掉元素名称。下一节将解释具体怎么做。

提示 通配符 *（星号）匹配代码中的任何元素名称。例如，使用 `* { border: 2px solid green; }` 会让每个元素都有一个 2 像素宽、绿色的实线边框！

提示 可以在一个选择器中使用一组元素名称，名称之间用逗号分隔。更多细节参见 9.8 节。

9.3 按类或 ID 选择元素

如果已经在元素中标识了 class（参见图 9.3.1）或 id（参见第 3 章），就可以在选择器中使用这个标准，从而只对被标识的元素进行格式化（参见图 9.3.2）。代码的生成结果如图 9.3.3 所示。

1. 按类选择要格式化的元素

(1) 输入 .（点号）。

(2) 不加空格，直接输入 *classname*，这里的 *classname* 标识希望应用样式的类。

2. 按 id 选择要格式化的元素

(1) 输入 #（井号）。

(2) 不加空格，直接输入 *id*，这里的 *id* 唯一标识希望应用样式的元素。

```
...
<article id="gaudi" class="about">
    <h1>Antoni Gaudí</h1>

    <p>Many tourists are drawn to Barcelona to see Antoni Gaudí's incredible architecture.</p>
    <p>Barcelona <a href="http://www.gaudi2002.bcn.es/english/" rel="external">celebrated the 150th
    ➝ anniversary</a> of Gaudí's birth in 2002.</p>

    <section class="project">
       <h2 lang="es">La Casa Milà</h2>
       <p>Gaudí's work was essentially useful. <span lang="es">La Casa Milà</span> is an apartment
       ➝ building and <em>real people</em> live there.</p>
    </section>

    ...
</article>

<p>This paragraph doesn't have <code>class="about"</code>, so it isn't red when the CSS is
➝ applied.</p>

<article class="about">
    <h1>Lluís Domènech i Montaner</h1>

    <p>Lluís Domènech i Montaner was a contemporary of Gaudí.</p>
    ...
</article>
...
```

图 9.3.1　有两个 class 为 about 的 article 元素。它们之间还有一个不含 class 的短段落

```
.about {
    color: red;
}
```

图 9.3.2　这个选择器会选择 class 为 about 的元素。在这个例子中，它们都是 article 元素，不过，可以将类应用于任何元素。如果只想对有这个类的 article 元素应用这种样式，可以将选择器写为 article.about。不过，这样做通常会使特殊性高于你所需要的程度

提示　可以单独使用 class 和 id，也可以同其他选择器标准混在一起使用。例如，.news { color: red; } 会影响所有属于 news 类的元素，而 h1.news { color: red; } 只会影响属于 news 类的 h1 元素。除非必须特别针对目标元素，最好不要在 id 或 class 选择器中添加元素名称。

提示　注意，在图 9.3.1 和图 9.3.2 中使用了表达内容含义的 class 名称（about），而不是命名为 red。最好避免创建描述事物如何显示的 class 名称，因为你可能在将来改变样式（例如，在本节示例中让文本显示为绿色）。同元素一样，类也会增加 HTML 的语义价值。

提示　如果图 9.3.2 中写的是 #gaudi { color: red; }，就只有第一个 article 的文本会显示为红色，因为它是唯一包含 id="gaudi" 的元素。每个 id 都必须是唯一的，因此，不能在关于 Lluís Domènech i Montaner 的 article 上再用这个 id。

提示　关于在 HTML 代码中为元素指定类的信息，参见 3.15 节。



图 9.3.3　属于 about 类的 article 显示为红色，而页面末尾的 p 元素没有显示为红色（或许你对链接显示为蓝色感到疑惑。链接显示为蓝色是浏览器的默认样式，可以编写自己的样式覆盖它。）（另见彩插）

类选择器与 ID 选择器的比较

要在 class 选择器和 id 选择器之间作出选择的时候，建议尽可能地使用 class 选择器。这主要是因为 class 选择器是可再用的。有人提议完全不使用 id 选择器，这可以理解，但最终的决定权掌握在网站开发人员手中。这是一个有争议的主题，双方都有一些说服力很强的观点。在任何情况下，id 选择器都会引入下面两个问题。

- 与它们关联的样式不能在其他元素上复用（记住，在一个页面中，一个 id 只能出现在一个元素上）。这会导致在其他元素上重复样式，而不是通过 class 共享样式。
- 它们的特殊性比 class 选择器要强得多。这意味着如果要覆盖使用 id 选择器定义的样式，就要编写特殊性更强的 CSS 规则。如果数量不多，可能还不难管理。如果你是在处理一个规模较大的网站，你的 CSS 就会变得比实际所需的更长、更复杂。

随着你越来越多地处理 CSS，你对这两点的理解也会更加清晰。（另一方面，有人喜欢使用 id 选择器的原因之一就是使用它们一眼就能看出元素是唯一的。）

因此，推荐的做法是寻找能将共享样式结合进一个或多个 class 的机会，从而可以对它们进行复用，同时，如果确实要使用 id 选择器，其数量也应维持在最小值（参见第 11 章的示例页面）。这样，样式表会变得更短，更易于管理。

9.4　按上下文选择元素

在 CSS 中，可以根据元素的祖先、父元素或同胞元素来定位它们（参见 1.3 节中的“父元素和子元素”），如图 9.4.1 ~ 图 9.4.4 所示。

祖先（ancestor）是包含目标元素（后代，descendant）的任何元素，不管它们之间隔了多少代。

1. 按祖先元素选择要格式化的元素

(1) 输入 *ancestor*，这里的 *ancestor* 是希望格式化的元素的祖先元素的选择器。

(2) 输入一个空格。

(3) 如果需要，对后续的每个祖先元素重复第 (1) 步和第 (2) 步。

(4) 输入 *descendant*，这里的 *descendant* 是希望格式化的元素的选择器。

```
...
<article class="about">
    <h1>Antoni Gaudí</h1>

    <p>Many tourists ... </p>
    <p>Barcelona ... </p>

    <section class="project">
       <h2 lang="es">La Casa Milà</h2>
       <p>Gaudí's work ... </p>
    </section>

    <section class="project">
       <h2 lang="es">La Sagrada Família</h2>
       ...
    </section>
</article>
...
```

图 9.4.1　为了让元素之间的关系更为明显，我对文本做了删节。每一级缩进代表一代。注意，在这段代码中，有两个直接包含在 article 里的属于 about 类的第二代 p 元素，一个直接包含在属于 project 类的 section（包含在 article 里）的第三代 p 元素。此外，还有一个第三代 p 元素，但没有显示出来。h2 也都是第三代。代码实现参见图 9.4.6

```
article.about p {
    color: red;
}
```

图 9.4.2　article.about 和 p 之间的空格意味着这个选择器会寻找任何作为 about 类的 article 的后代的 p 元素，不管它是第几代元素。不过，在 class 前面添加元素名称通常会带来更高的特殊性，而在实践中往往并不需要这么高的特殊性（对 id 来说更是如此）。关于特殊性较低的选择器，参见图 9.4.3

```
/* 得到相同效果的其他方式
------------------------------------ */

/* 任意article后代的所有p，
→特殊性最低的方法 */
article p {
    color: red;
}

/* 任意about类元素后代的所有p，
→特殊性第二低的方法 */
.about p {
    color: red;
}
```

图 9.4.3　构造选择器以达到预期效果通常有一种以上的方法。关键在于你需要多大的特殊性。这里第一个例子中的选择器（article p { }）的特殊性比它后面的选择器（.about p { }）和图 9.4.2 中选择器的特殊性都低。这里第二个例子结合了 class 选择器和后代选择器（也可以与 id 选择器结合）。你将发现一直使用的是这些选择器，而不是图 9.4.2 中更特殊和冗长的模式



图 9.4.4　所有包含在属于 about 类的元素里的 p 元素都是红色的，即使它们同时也包含于其他属于 about 类的元素。图 9.4.2 和图 9.4.3 中每条样式规则的结果都显示在这里（另见彩插）

9

提 示　基于元素祖先的选择器称为后代选择器。CSS3 将其重命名为**后代结合符**（有的人仍称其为“选择器”）。

提 示　不要对图 9.4.2 中的 article.about 部分感到疑惑。记住这只表示“class 等于 about 的 article”。因此 article.about p 表示“包含在 class 等于 about 的 article 元素里的任何 p 元素”。相比之下，特殊性低一些的 .about p 表示“包含在 class 等于 about 的任意元素里的所有 p 元素”，参见图 9.4.3。这是因为在上下文中 id 选择器的特殊性比元素和类选择器的更高。

上面的例子展示了后代结合符。CSS 也有子结合符，从而可以为父元素的直接后代（即子元素）定义样式规则。在 CSS3 之前，它们称做子选择器。父元素是直接包含另一元素（子元素）的元素，也即它们之间没有隔着任何层次。

2. 按父元素选择要格式化的元素

(1) 输入 *parent*，这里的 *parent* 是直接包含待格式化元素的元素的选择器。

(2) 输入 >（大于号），参见图 9.4.5。

(3) 如果需要，对后续每代父元素重复第 (1) 步和第 (2) 步。

(4) 输入 *child*，这里的 *child* 是要格式化的元素的选择器。

```
article.about > p {
    color: red;
}
```

图 9.4.5　这个选择器仅选择作为 about 类 article 元素的子元素（而非子子元素、子子子元素等）的 p 元素。包含于任何其他元素的 p 元素均不会被选中，实现参见图 9.4.6



图 9.4.6　只有头两个 p 元素是 about 类 article 的子元素。另两个 p 元素是 article 里 section 元素的子元素。这个例子使用的 HTML 代码见图 9.4.1（另见彩插）

提 示　正如你在后代结合符中看到的，可以忽略 class 前面的元素名称。事实上，这也是推荐的做法（除非为实现目标样式需要额外的特殊性）。例如，使用 .about > p { color: red; } 会产生同样的效果。也可以完全不用 class，如 article > p { color: red; }，产生更低的特殊性。在考虑特殊性更高的选择器之前，应当尽可能地使用这些更为简单的形式。本章剩余部分的一些例子也可以按类似的方式进行简化。你已经了解了如何进行简化，因此不必再一一列举这些替代方案，但应记住，通常最好保持**较低**的特殊性，让样式更易于复用。

提 示　也可以在子结合符中使用 id 选择器，但推荐尽量使用特殊性更低的选择器（如元素类型、class 等）。

提 示　Internet Explorer 6 不支持子选择器。

有时，只选择元素的第一个子元素（而不是所有的子元素）是很有用的。要实现这种效果，可以使用 :first-child 伪类（参见图 9.4.7 ~ 图 9.4.10）。

```
/* 你可能认为这会让第一个段落变成红色，
→但实际上不会！ */

.about > p:first-child {
    color: red;
}
```

图 9.4.7　:first-child 伪类仅选择某元素的第一个子元素，而不是作为子元素的元素的第一个实例。因此，也许你倾向于认为下面的规则会让示例页面中的第一个段落显示为红色，但实际上不是这样（参见图 9.4.8），因为 h1 才是 about 类 article 的第一个子元素。这个例子使用的 HTML 代码见图 9.4.1

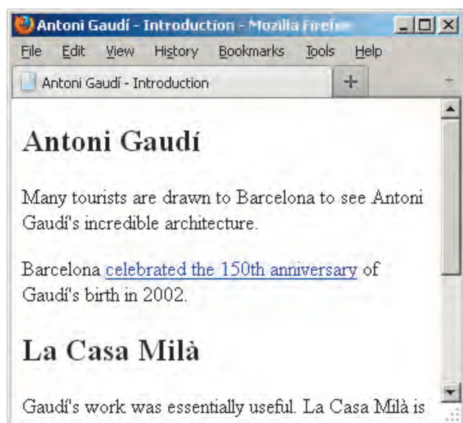


图 9.4.8　这条规则对页面没有任何影响，因为没有一个 p 元素是 about 类元素的子元素

```
/* h1是第一个子元素，因此是有效的 */

.about > h1:first-child {
    color: red;
}
```

图 9.4.9　这个选择器仅选择作为在 HTML 中带有 class="about" 的元素的第一个子元素的 h1。这条规则会影响页面的显示，参见图 9.4.10



图 9.4.10　包含在 article 里的 h1 显示为红色，因为它是某个 about 类元素的第一个子元素。如果 article 里还有其他 h1 元素，这些元素将不会显示为红色（另见彩插）

3. 选择某元素的第一个子元素进行格式化

(1) 这一步可选，输入 *parent*，这里的 *parent* 是目标元素的父元素的选择器。

(2) 如果在第 (1) 步中包含了 parent，就再依次输入一个空格、> 和另一个空格。

(3) 可选，输入表示要格式化的第一个子元素的选择器（如 p 或 .news）。

(4) 输入 :first-child（参见图 9.4.9）。

（注意，第 (1) 步中指定 parent 并不是必需的。例如，使用 **p:first-child { font-weight: bold; }** 会为每一个作为某元素第一个子元素的段落添加粗体样式。）

下面继续讲我们的“家族”主题。同胞元素（sibling）是拥有同一父元素的任何类型的子元素。相邻同胞元素（adjacent sibling）是直接相互毗邻的元素，即它们之间没有其他的同胞元素。在下面这个简略的例子中，h1 和 p 是相邻同胞元素，p 和 h2 是相邻同胞元素，而 h1 和 h2 则不是相邻同胞元素。不过，它们都是同胞元素（也是 body 元素的子元素）。

```
…
<body>
```

```
        <h1>...</h1>
        <p>...</p>
        <h2>...</h2>
    </body>
    </html>
```

CSS 相邻同胞结合符（adjacent sibling combinator）可以选择直接跟在指定的同胞元素后面的同胞元素。（关于 CSS3 中新出现的普通同胞结合符（general sibling combinator），参见最后一条提示。）

4. 按相邻同胞元素选择要格式化的元素

(1) 输入 *sibling*，这里的 *sibling* 是包含在同一父元素中的、直接出现在目标元素前面的元素的选择器。（它们不必是同一种元素类型，只要它们彼此直接相邻就行。）

(2) 输入 +（加号）。

(3) 如有需要，对每个后续的同胞元素重复第 (1) 步和第 (2) 步。

(4) 输入 *element*，这里的 *element* 是要格式化的元素的选择器，参见图 9.4.11。

```
.about p+p {
    color: red;
}
```

图 9.4.11　这个相邻同胞结合符仅选择直接出现在同胞 p 元素后面的 p 元素

提 示　另参见第 1 章“父元素和子元素”。

提 示　选择器中的 :first-child 部分称为伪类，因为它标识的是你（设计人员或开发人员）无需在 HTML 代码中标记的一组元素。

提 示　IE 6 既不支持 :first-child，也不支持相邻同胞选择器。

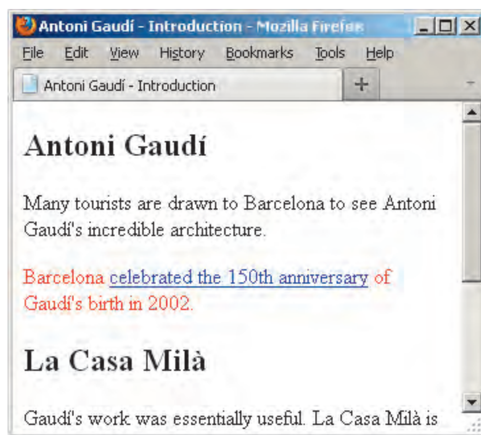


图 9.4.12　只有直接跟在同胞 p 元素后面的 p 元素显示为红色。如果后面还有第三个、第四个以及更多的段落，它们也将显示为红色。例如，如果要对除第一个段落以外的所有段落进行缩进，相邻同胞结合符就很有用（另见彩插）

提 示　CSS3 引入了普通同胞结合符，通过它可以选择不一定直接出现在另一同胞元素后面的同胞元素。它与相邻同胞结合符的唯一区别是使用 ~（波浪号）代替 + 分隔同胞元素。例如，**h1 ~ h2** { color: red; } 会让任何属于同一父元素的同胞 h1 后面的 h2 元素显示为红色（它们可以直接相邻，也可以不直接相邻），参见图 9.4.12。

9.5　选择元素的一部分

可以只选择元素的第一个字母或第一行，并对其添加样式。

1. 选择元素的第一行

(1) 输入 *element*，这里的 *element* 是要对其第一行进行格式化的元素的选择器。

(2) 输入 :first-line 以选择第一步中引用的元素的第一整行，参见图 9.5.1、图 9.5.2 和图 9.5.3。

2. 选择元素的第一个字母

(1) 输入 *element*，这里的 *element* 是要对其第一个字母进行格式化的元素的选择器。

(2) 输入 :first-letter 以选择第 (1) 步中引用的元素的第一个字母，参见图 9.5.4 和图 9.5.5。

```
<article class="about">
    <h1>Antoni Gaudí</h1>

    <p>Many tourists are drawn to Barcelona
    → to see Antoni Gaudí's incredible
    → architecture.</p>
    <p>Barcelona <a href="http://
    → www.gaudi2002.bcn.es/english/"
    → rel="external">celebrated the 150th
    → anniversary</a> of Gaudí's birth in
    → 2002.</p>

    <section class="project">
       <h2 lang="es">La Casa Milà</h2>
       <p>Gaudí's work was essentially
       → useful. <span lang="es">La Casa
       → Milà</span> is an apartment
       → building and <em>real people</em>
       → live there.</p>
    </section>

    <section class="project">
       <h2 lang="es">La Sagrada Família</h2>
       <p>The complicatedly named and
       → curiously unfinished Expiatory
       → Temple of the Sacred Family is the
       → <em>most visited</em> building in
       → Barcelona.</p>
    </section>
</article>
```

图 9.5.1 这里无法分辨哪些文字会受到 first-line 的影响，因为只有在浏览器中查看页面，根据内容的流动，才能看出哪些文字在第一行。这里讲的第一行并不是由 HTML 中的分行决定的

```
p:first-line {
    color: red;
}
```

图 9.5.2 这个选择器会选择每个 p 元素的第一行

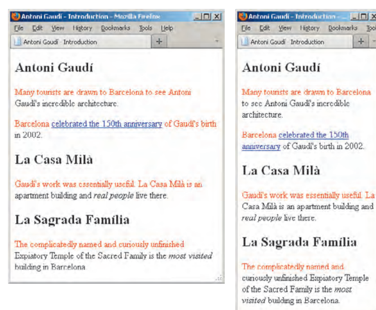


图 9.5.3 调整浏览器的宽度就会改变第一行的内容（并会改变受格式化影响的范围）

```
p:first-letter {
    color: red;
}
```

图 9.5.4 这个选择器仅会选择每个 p 元素的第一个字母。对应的 HTML 代码见图 9.5.1

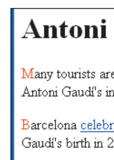


图 9.5.5 first-letter 可用于创建首字母下沉的效果（但要等讲到除 color 以外的更多属性以后）

提 示 根据 CSS 规范，选择器应包含第一个字母前面的标点符号。现代浏览器都支持这一特性，但旧版本的 IE 却并不是这样做的，它将标点符号本身当做第一个字母。

提 示 只有某些特定的 CSS 属性可以应用于 :first-letter 伪元素，包括 font、color、background、text-decoration、vertical-align（只要 :first-letter 不是浮动的）、text-transform、line-height、margin、padding、border、float 和 clear。这些属性将在第 10 章和第 11 章讲到。

提示　可以将 :first-letter 和 :first-line 伪元素与比这个例子中的选择器更复杂的选择器结合使用。例如，如果要选择包含在 project 类元素中的每个段落的第一个字母，可以使用 .project p:first-letter。

伪元素、伪类及 CSS3 的 ::first-line 和 ::first-letter 语法

在 CSS3 中，**:first-line** 的语法为 **::first-line**，**:first-letter** 的语法为 **::first-letter**。注意，它们用两个冒号代替了单个冒号。

这样修改的目的是将伪元素（有四个，包括 ::first-line、::first-letter、::before 和 ::after）与伪类（如 :first-child、:link、:hover 等）区分开。

伪元素（pseudo-element）是 HTML 中并不存在的元素。例如，定义第一个字母或第一行文字时，并未在 HTML 中作相应的标记。它们是另一个元素（在本例中为 p 元素）的部分内容。

相反，**伪类**（pseudo-class）则应用于 HTML 元素。例如，使用 :first-child 可以选择某元素的第一个子元素。

未来，::first-line 和 ::first-letter 这样的双冒号语法是推荐的方式，现代浏览器也支持它们。原始的单冒号语法则被废弃了，但浏览器出于向后兼容的目的，仍然支持它们。不过，IE9 之前的 Internet Explorer 版本均不支持双冒号。因此，你可以选择继续使用单冒号语法，除非你为 IE8 及以下版本设置了单独的 CSS。

9.6　按状态选择链接元素

CSS 允许根据链接的当前状态对它们进行格式化。链接的状态包括访问者是否将鼠标停留在链接上，链接是否被访问过，等等。可以通过一系列伪类实现这一特性（如图 9.6.1 和图 9.6.2 所示）。

```
...
    <p>Many tourists are drawn to Barcelona
    → to see Antoni Gaudí's incredible
    → architecture.</p>
    <p>Barcelona <http://www.gaudi2002.
    → bcn.es/ english/">celebrated</a> the
    → 150th anniversary of Gaudí's birth
    → in 2002.</p>
...
```

图 9.6.1　无法在代码中指定链接的状态。链接的状态是由访问者控制的。伪类让你可以获取链接的状态，以改变链接在该状态下显示的效果

```
a:link {
    color: red;
}

a:visited {
    color: orange;
}

a:focus {
    color: purple;
}

a:hover {
    color: green;
}

a:active {
    color: blue;
}
```

图 9.6.2　链接的样式应该始终按照这种顺序进行定义，以避免链接处于多种状态（如已访问和鼠标停留）时覆盖属性

按状态选择要格式化的链接元素的步骤

(1) 输入 a（a 是链接元素的名称）。

(2) 输入 :（冒号）。

(3) 输入 link 以设置从未被激活或指向，当前也没有被激活或指向的链接的外观（如图 9.6.3 所示）。

或者输入 visited 以设置访问者已激活过的链接的外观（如图 9.6.4 所示）。

或者输入 focus，如果链接是通过键盘选择并已准备好激活的（如图 9.6.5 所示）。

或者输入 hover 以设置正被指向的链接的外观（如图 9.6.6 所示）。

或者输入 active 以设置激活过的链接的外观（如图 9.6.7 所示）。

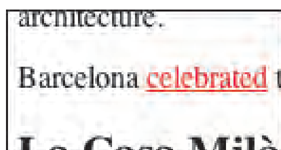


图 9.6.3　新的、未访问的链接显示为红色（另见彩插）



图 9.6.4　访问过的链接变为橙色（另见彩插）

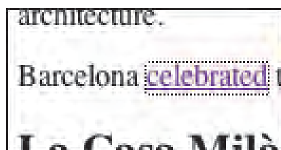


图 9.6.5　链接获得焦点（如通过 Tab 键）时显示为紫色（另见彩插）

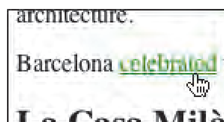


图 9.6.6　当访问者将鼠标指针停留在链接上时，它显示为绿色（另见彩插）



图 9.6.7　当访问者激活链接时，它变为蓝色（另见彩插）

提示　也可以对其他类型的元素使用 :active 和 :hover 伪类。例如，设置 **p:hover { color: red; }** 以后，当鼠标停留在任何段落上时，段落都会显示为红色。（注意，这一特性仅对 Internet Explorer 7 及以上版本有效，对 IE6 无效。同时，IE6 和 IE7 均不支持对 a 使用 :active。其他浏览器对这两个特性都支持。）

提示　由于链接可能同时处于多种状态（如同时处于激活和鼠标停留状态），且晚出现的规则会覆盖前面出现的规则，所以，一定要按照下面的顺序定义规则：link、visited、focus、hover、active（缩写为 LVFHA）。一种助记口诀为“Lord Vader’s Former Handle Anakin”（达斯·维达的原名叫安纳金）。有人提议使用 LVHFA 的顺序。这也是可行的。

9

9.7　按属性选择元素

可以对具有给定属性或属性值的元素进行格式化（参见图 9.7.1、图 9.7.2 和图 9.7.3）。

```
<article class="about">
    <h1>Antoni Gaudí</h1>

    <p>Many tourists are drawn to Barcelona to see Antoni Gaudí's incredible architecture.</p>
    <p>Barcelona <a href="http://www.gaudi2002.bcn.es/english/" rel="external">celebrated the 150th
    ➝ anniversary</a> of Gaudí's birth in 2002.</p>

    <section class="project">
       <h2 lang="es">La Casa Milà</h2>
       <p>Gaudí's work was essentially useful. <span lang="es">La Casa Milà</span> is an apartment
       ➝ building and <em>real people</em> live there.</p>
    </section>

    <section class="work">
       <h2 lang="es">La Sagrada Família</h2>
       <p>The complicatedly named and curiously unfinished Expiatory Temple of the Sacred Family is
       ➝ the <em>most visited</em> building in Barcelona.</p>
    </section>
</article>
```

图 9.7.1　出于演示的目的，将第二个 section 元素的 class 值由 project 改成了 work

```
section[class] {
    color: red;
}
```

图 9.7.2　方括号包围目标属性和目标属性值。这个例子中没有属性值，它选择的是所有具有 class 属性的 section



图 9.7.3　每个包含 class 属性的 section 元素（无论其 class 值是什么）都显示为红色（另见彩插）

按属性选择要格式化的元素的步骤

(1) 输入 *element*，这里的 *element* 是要考察其属性的元素的选择器。

(2) 输入 [*attribute*，这里的 *attribute* 是选择元素需要考察的属性的名称。

(3) 输入 ="*value*"，表示属性值等于这里的 *value* 的元素将被选中。或者输入 ~="*value*"，表示属性值包含这里的 *value* 的元素将被选中（属性值还可以包含其他内容，不同的属性值之间用空格分隔）。它必须匹配完整的单词，而不是单词的一部分。或者输入 |="*value*"（前面是管道符号，而不是数字 1 或小写字母 l），表示属性等于这里的 *value* 或以 *value* 开头（即你输入的值后面加上连字符）的元素将被选中。（这常用以搜索包含 lang 属性的元素。）

或者输入 ^="*value*"，表明属性值以这里的 *value* 开头（作为完整的单词，或单词的一部分）的元素将被选中（这是 CSS3 中新增的特性，参见本节提示）。

或者输入 $="*value*"，表明属性值以这里的 *value* 结尾（作为完整的单词，或单词的一部分）的元素将被选中（这是 CSS3 中新增的特性，参见本节提示）。

或者输入 *="*value*"，表明属性值至少包含这里的 *value* 一次的元素将被选中。也就是说，*value* 不必是属性值中的完整单词（这是 CSS3 中新增的特性，参见本节提示）。

(4) 输入]。

提示 当前所有主流浏览器（包括 IE7）均支持按元素包含的属性（和属性值）选择元素。对于第 (3) 步中提到的 CSS3 中新增的属性选择器，IE7 和 IE8 有一些异常。更多信息参见 http://reference.sitepoint.com/css/css3attribute-selectors。

更多属性选择器示例

属性选择器非常强大。这里给出更多的例子，演示属性选择器几种不同的使用方式。这些例子在实践中也用得上。

- 这个选择器选择任何 rel 属性值等于 external（必须完全匹配）的 a 元素。

```
a[rel="external"] {
   color: red;
}
```

- 假设某 section 元素有两个类，如 <section **class="project barcelona"**>，另一个 section 有一个类，如 <section **class="barcelona"**>。~ = 语法可以测试单词的部分匹配，即匹配出现在以空格相隔的单词列表中的完整的单词。在这个例子中，两个元素都将显示为红色。

```
section[class ~ ="barcelona"] {
  color: red;
}

/* 这个选择器也能匹配，因为这个选择器匹配部分字符串（不需要完整的单词） */

section[class*="barc"] {
  color: red;
}

/* 这个选择器无法匹配，因为 barc 并不是空格分隔的列表中的某个完整单词 */

section[class ~ ="barc"] {
  color: red;
}
```

- 这个选择器选择任何带有 lang 属性且属性值以 es 开头的 h2。在 HTML 代码示例（参见图 9.7.1）中有两个这样的实例。

9

```
h2[lang|="es"] {
  color: red;
}
```

- ❑ 通过使用通用选择器，这个选择器选择任何带有 lang 属性且属性值以 es 开头的元素。在 HTML 代码示例（参见图 9.7.1）中有三个这样的实例。

```
*[lang|="es"] {
  color: red;
}
```

- ❑ 通过联合使用多种方法，这个选择器选择所有既有任意 href 属性，又有任意属性值包含单词 howdy 的 title 属性的 a 元素。

```
a[href][title ~ ="howdy"] {
  color: red;
}
```

- ❑ 作为上一选择器的精确度低一些的变体，这个选择器选择所有既有任意 href 属性，又有任意属性值包含 how（作为完整的单词或单词的一部分，它匹配 how、howdy、show 等，无论 how 出现在属性值的什么位置）的 title 属性的 a 元素。

```
a[href][title*="how"] {
  color: red;
}
```

- ❑ 这个选择器匹配任何 href 属性值以 http:// 开头的 a 元素。

```
a[href^="http://"] {
  color: orange;
}
```

- ❑ 这个选择器匹配任何 src 属性值完全等于 logo.png 的 img 元素。

```
img[src="logo.png"] {
  border: 1px solid green;
}
```

- ❑ 这个选择器的精确度比前一个低一些，它匹配任何 src 属性值以 .png 结尾的 img 元素。

```
img[src$=".png"] {
  border: 1px solid green;
}
```

这些例子并不代表你能实现的所有效果，不过，希望它们可以激发你继续探索。

9.8　指定元素组

经常需要将相同的样式规则应用于多个元素。可以为每个元素重复地设置样式规则，也可以组合选择器，一次性地设置样式规则。当然，后一种方法效率更高，通常也会让样式表更易于维护。

将样式应用于元素组的步骤

(1) 输入 *selector1*，这里的 *selector1* 是受样式规则影响的第一个元素的名称。

(2) 输入 ,（逗号）。

(3) 输入 *selector2*，这里的 *selector2* 是受样式规则影响的下一个元素的名称。

(4) 对其他每个元素重复第 (2) 步和第 (3) 步。具体示例参见图 9.8.1。

```
...
<article id="gaudi" class="about">
    <h1>Antoni Gaudí</h1>

    <p>Many tourists are drawn ...</p>
    <p>Barcelona ...</p>

    <section class="project">
       <h2 lang="es">La Casa Milà</h2>
       <p>Gaudí's work was ...</p>
    </section>

    <section class="project">
       <h2 lang="es">La Sagrada Família
       → </h2>
       <p>The complicatedly named ...</p>
    </section>
</article>
...
```

图 9.8.1　这段代码包含一个 h1 和两个 h2

提 示　通过元素组添加样式只是一种简写方式。h1, h2 { color: red; } 这个规则与 h1 { color: red; } 和 h2 { color: red; } 这两个规则的效果完全一样，如图 9.8.2 和图 9.8.3 所示。

```
h1,
h2 {
    color: red;
}
```

图 9.8.2　可以列出任意数量的单独的选择器（无论它们包含的是元素名称、id 还是 class），只要用逗号分隔它们。不一定像示例代码那样让每个选择器位于单独的行，不过很多开发人员坚持这样做，以增强代码的可读性。当选择器很长时，这样做的好处更明显



图 9.8.3　通过同一条样式规则，可以让 h1 和 h2 元素都显示为红色（另见彩插）

提 示　可以组合使用任何类型的选择器，从最简单的（如图 9.8.2 所示）到最复杂的都可以。例如，可以使用 h1, .project p:first-letter 来选择一级标题以及包含在 class 等于 project 的所有元素中的 p 元素的第一个字母。

提示 有时，为可应用于多个选择器的共同样式创建一个样式规则，再为没有共同点的样式分别创建单独的样式规则是很有用的。要记住，在样式表中后指定的规则会覆盖先指定的规则（参见 7.3 节）。

9.9 组合使用选择器

本章的示例都尽量保持简单，以便你全面了解各种类型的选择器。不过，现实中常常需要组合使用这些技术，才能找到要格式化的元素，但这也是选择器的功能强大之所在。

图 9.9.1（实现结果见图 9.9.2）通过一个极端的例子展示了如何组合使用选择器。这里展示了几种实现同样效果的不同方式，它们是按特殊性由低向高排列的。

```
em {
    color: red;
}

.project em {
    color: red;
}

.about .project em {
    color: red;
}

#gaudi em {
    color: red;
}
```

上面都是你每天编写 CSS 经常用到的典型的选择器类型（不过，前面提到过，最好尽量少用 id 选择器）。对大多数设计来说，无论它们看起来有多复杂，都用不着编写那些近乎疯狂的选择器。

```
.project h2[lang|="es"] + p em {
    color: red;
}
```

图 9.9.1 这对你来说可能是个挑战。从右向左看，它表明“仅选择 em 元素，它们包含在 p 元素中，这样的 p 元素是 lang 属性值以 es 开头的 h2 元素的直接相邻同胞元素，且是 class 等于 project 的任何元素的子元素”。弄明白了吗？实际上，你很少需要编写这么复杂的选择器，但至少你需要知道，必要时可以这样做。或者，当你想吓唬阅读你代码的人时可以这样做



图 9.9.2 图 9.9.1 中所有那些东西只是让 em 元素显示为红色吗？！如果你认为简单地写成 **.about em { color: red; }** 要好得多（也更易于维护），你肯定是对的。除非你需要特别高的特殊性，否则不必使用这些吓人的玩意（另见彩插）

因此，必要的时候可以组合使用选择器，不过，最好将特殊性控制在刚好需要的程度。例如，如果你只想选择包含在 `class="project"` 的元素内的 em 元素，可以使用 **`.project em { color: red; }`**。尽管在 HTML 中 em 元素是嵌套在 p 元素中的，但没有必要写成 `.project p em { color: red; }`，除非你不想为段落以外的 em 元素应用该样式。总之，从最简单的开始，按需增加特殊性。

更多 CSS3 选择器

CSS3 为你的工具箱增加了不少新的选择器。你已经在本章中见到了其中的一些。其他的新选择器大多为伪类，其中的一些还相当复杂，不过也非常强大。所有 CSS3 选择器及其完整描述见 www.w3.org/TR/css3-selectors/#selectors，简介与示例参见 www.w3.org/wiki/CSS/Selectors，浏览器支持情况参见 http://findmebyip.com/litmus。你将看到，除了 Internet Explorer 以外，其他浏览器的支持程度都很好。Internet Explorer 直到 IE9 才开始支持大多数 CSS3 中新增的选择器（尤其是伪类和伪元素）。

9.10 选择器回顾

回顾一下，我们讨论了以下类型的选择器（它们均可组合使用）：

- 按上下文选择元素
- 按名称选择元素
- 按 class 或 id 选择元素
- 通过伪类或伪元素选择元素
- 根据属性选择元素

9

第 10 章

文本的格式化

10

本章内容

- ❑ 选择字体系列
- ❑ 指定替代字体
- ❑ 创建斜体
- ❑ 应用粗体格式
- ❑ 设置字体大小
- ❑ 设置行高
- ❑ 同时设置所有字体值
- ❑ 设置颜色
- ❑ 修改文本的背景
- ❑ 控制间距
- ❑ 增加缩进
- ❑ 设置空白属性
- ❑ 对齐文本
- ❑ 修改文本的大小写
- ❑ 使用小型大写字母
- ❑ 装饰文本

使用 CSS 可以修改文本的字体、大小、粗细、倾斜、行高、前景和背景颜色、间距和对齐方式，可以决定是否为文本添加下划线或删除线，可以将文本转化为全部使用大写字母、全部使用小写字母或使用小型大写字母。而且，通过短短几行代码就可以让这些样式应用于整篇文档或整个网站。在本章中，你将学习如何做到这些。

尽管本章中讨论的很多属性主要是应用于文本的，但这并不意味着它们只能应用于文本。其中很多属性也可以应用于其他类型的内容。图 10.0.1 是一个没有应用任何样式表的页面，本章接下来会将其各部分的文本格式化。

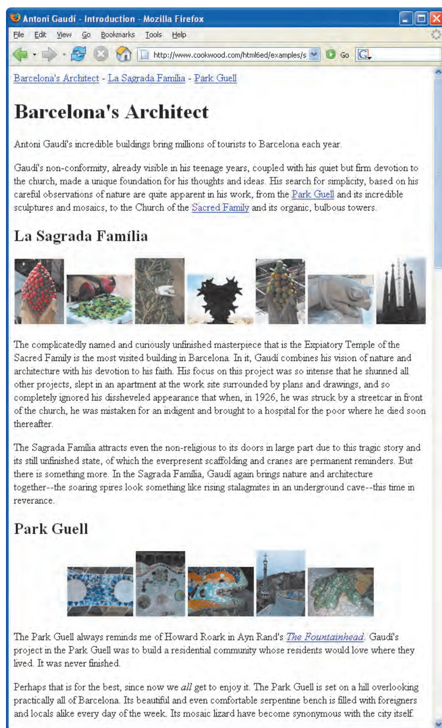


图 10.0.1　这是一个没有应用任何样式表的页面的样子。（在不同的浏览器中，默认的标题字号可能不一样。）可以在本书网站的示例部分找到它的 HTML 源代码（以及接下来的 CSS 示例）：www.bruceontheloose.com/htmlcss/examples/（另见彩插）

第 11 章会接着讨论 CSS 布局。

10.1 选择字体系列

对于自己的网站，一个很重要的选择就是选择标题和主体文本所用的字体。你将了解到，并非所有的系统都支持相同的默认字体，因此，应该定义替代字体作为备选。不过，让我们首先看看如何定义单个字体（如图 10.1.1 和图 10.1.2 所示），以及未提供替代字体的影响（如图 10.1.3 所示）。

```
body {
    font-family: "Palatino Linotype";
}

h1, h2 {
    font-family: "Arial Black";
}
```

图 10.1.1 由于对 body 元素设置了 Palatino Linotype 字体，该设置会传递给其他元素。通过对 h1 和 h2 元素设置 Arial Black 字体，上述设置被覆盖了。不过，你即将看到，仅仅定义单个字体是不够的，因为并非所有操作系统都支持这种字体。示例中的 Palatino Linotype 是 Windows 上的常见字体，但 Mac OS 或 Linux 系统上可能没有这种字体

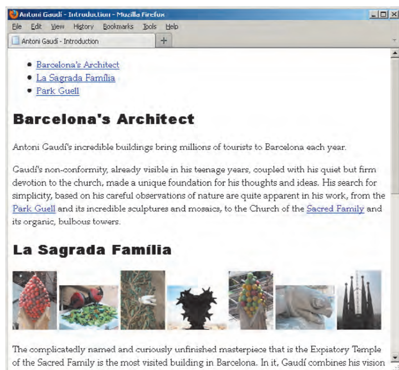


图 10.1.2 在此 Windows 系统中，Palatino Linotype 字体已经安装，可以正常显示。如你所见，body 的 font-family 设置传递给了 a 和 p 元素。如果没有为 h1 和 h2 指定 Arial Black 字体，它们也将显示为 Palatino Linotype 字体



图 10.1.3 一些 Mac 系统中并未安装 Palatino Linotype 字体。如果选择了访问者系统中没有的字体，他们的浏览器中就会显示默认字体（如图所示，在这个例子中是 Times 字体）

设置字体的方法

在样式表中需要的选择器之后，输入 `font-family: name`，这里的 *name* 是首选字体的名称。

提示 对于包含多个单词的字体名称，应该用引号（单引号或双引号）包围起来。

提示 如果字体名称包含非 ASCII 字符，就必须声明样式表的编码，即在样式表的第一行添加 `@charset "UTF-8";`。实际上，即使当前没有这样做的必要，对所有样式表都这样做也没有坏处。这样做可以避免日后出现问题。

提示 可以指定想要设置的任何字体，不过访问者只会看到他们的系统里安装的字体。更多细节参见下一节。

提示 使用通用的 font 属性可以一次性定义字体、大小和行高。参见 10.7 节。

提示 font-family 属性是继承的。

10.2 指定替代字体

尽管开发人员或设计人员可以指定自己想要的任何字体，但是对于访问者来说，只有当他们的计算机上也安装了这种字体的时候，他们才能看见这种字体。因此，最好使用访问者系统上理应包含的那些字体。这里有一段 Windows 和 Mac OS 默认都包含的字体的列表（详细信息见“Mac OS 和 Windows 上默认共有的字体”）。

此外，还有其他需要考虑的问题。如果要指定的字体在两种操作系统上的名称不一样，就可以同时指定这两个名称，而操作系统会使用它安装了的那一个。类似地，如果要指定的字体只存在于一种操作系统，就可以为另一种操作系统选择一个替代字体。这两个字体可能并不完全一样，但这样做的目的是指定尽可能接近的字体。最后一点，最好指定一种表示类属的标准字体[①]，以防系统对所列的字体都不支持，参见图 10.2.1、图 10.2.2 和图 10.2.3。

指定替代字体的步骤

(1) 输入 `font-family: name`，这里的 *name* 是首选字体的名称。

(2) 输入 `, name2`，这里的 *name2* 是第二字体的名称。用一个逗号和空格分隔每个字体。

(3) 根据需要，重复第 (2) 步，最后以一个表示类属的标准字体（`serif`、`sans-serif`、`cursive`、`fantasy` 或 `monospace`[②]，它们代表首选字体最为接近的风格）结束字体列表。

```
body {
    font-family: "Palatino Linotype",
    → Palatino, serif;
}

h1,
h2 {
    font-family: "Arial Black", Arial,
    → sans-serif;
}
```

图 10.2.1 在 `font-family` 属性中可以包含替代字体。如果系统上没有安装首选字体，浏览器就会使用替代字体。在这个例子中，我们告诉浏览器，如果系统未安装 Palatino Linotype 字体，就寻找 Palatino 字体（参见图 10.2.3）；如果这两种字体都未安装，就退回到标准的 serif 字体。在 `font-family` 属性中声明的字体列表称为字体栈（font stack）。此外，标题也添加了替代字体

提 示 可以在同一个 `font-family` 规则中为不同的字母表指定字体（如日语和英语），从而对包含不同语言和书写体系的文本进行格式化。

提 示 系统通常都具有下列表示类属的字体名称对应的字体：`serif`、`sans-serif`、`cursive`、`fantasy` 和 `monospace`。因此，标准的做法是在字体栈的末尾指定上述字体名称中的一种，以防所有其他的字体都未安装。其中，`serif` 和 `sans-serif` 是（目前为止）用得最多的，因为它们分别对应于最为常见的两类字体。

① 表示类属的标准字体指的是 serif、sans-serif 等字体，它们表示的不是单个字体，而是一类字体，例如，serif 表示有衬线字体（在字的笔画开始、结束的地方有额外的装饰，且笔画的粗细有差异），sans-serif 表示无衬线字体（字的笔画无额外的装饰，粗细差不多）。——译者注

② `cursive`、`fantasy`、`monospace` 分别表示手写字体、装饰字体和等宽字体。——译者注

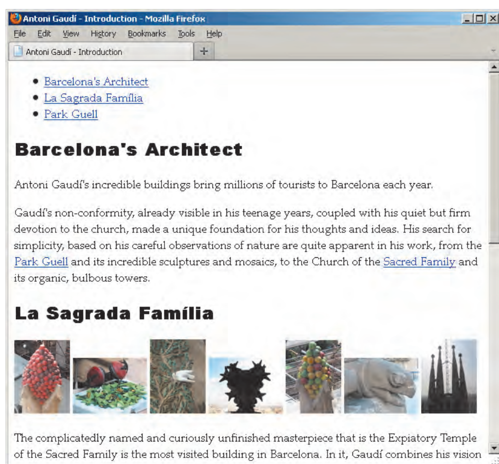


图 10.2.2　安装了 Palatino Linotype 字体的系统会继续使用该字体

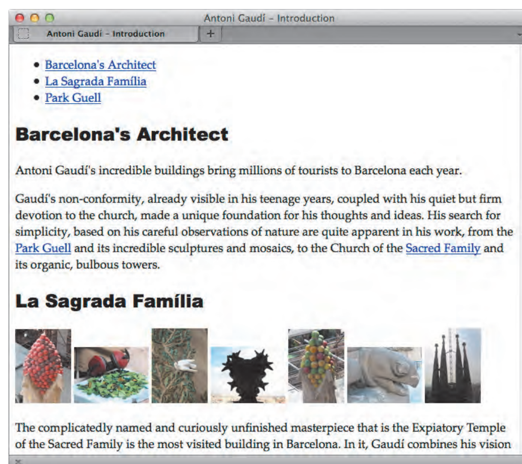


图 10.2.3　没有 Palatino Linotype 字体的系统会使用 Palatino 字体（只要该系统有这种字体；大多数 Mac 系统都有这种字体）。如果 Palatino 也没有，浏览器会试着寻找第三个选项。几乎所有的系统均包含 serif 类和 sans-serif 类字体。注意，不同字体默认的行高并不相同。稍后我们将对行高进行调整

Mac OS 和 Windows 上默认共有的字体

Mac OS 和 Windows 上默认都有的字体是非常有限的，它们仅包括 Arial、Arial Black、Comic Sans MS、Courier New、Georgia、Impact、Trebuchet MS、Times New Roman 和 Verdana。结果，绝大多数网站使用的字体都无外乎这些（其中 Arial 可能是用得最多的一个）。尽管它们在 Mac OS 和 Windows 上的浏览器中显示的效果也并不完全一致，但至少你可以肯定它们都能显示出来。

除此之外，还有别的选择。Mac OS 和 Windows 都包含了更多（却不相同的）可用于字体栈的系统字体。在网上搜索“font stacks”（字体栈），就可以看到很多 `font-family` 声明。可以将这些声明复制到你的样式表里，从而为不同的访问者提供相似的字体。

此外，还可以在网页中加载系统默认没有的字体，这种方式变得越来越普遍。第 13 章将讲解如何实现这种方式。

10.3　创建斜体

在传统出版业中，斜体通常用来表示引述、强调的文本、外文单词（如 *de rigueur*）、科学名词（如 *Homo sapiens*）、电影片名等。

浏览器通常让一些 HTML 元素（如 `cite`、`em` 和 `i`）默认以斜体显示，因此，不必在 CSS 中对这些元素设置斜体。正如 1.2 节中讲到的，HTML 用于描述内容的含义，而不是控制它们显示的样子。有时你需要让一些内

容以斜体显示，但不应该通过使用上述元素对它们进行标记以创建斜体。通过使用 CSS 的 font-style 属性，可以让任何元素中的文本以斜体显示。

作为示例，图 10.3.1 说明了如何对段落文本添加斜体样式。（由于这样显示难以阅读，因此，这一样式规则将在接下来的例子中去掉。）

```
body {
    font-family: "Palatino Linotype",
    → Palatino, serif;
}

h1,
h2 {
    font-family: "Arial Black", Arial,
    → sans-serif;
}

p {
    font-style: italic;
}
```

图 10.3.1　在这个例子中，段落以斜体显示

字体的斜体版本通常是由字体设计师从头创建的，尤其是有衬线字体。字体的斜体版本不仅仅是普通文本的倾斜版本，它们的形式有一些合理的差异。例如，Palatino Linotype 就有一份真正的斜体字体，参见图 10.3.2。通过字母 a 可以很清晰地看到，斜体不仅仅是对字母进行倾斜产生的效果。不过，有的字体并没有斜体版本。如果对这些字体的文本设置 font-style: italic，浏览器就会显示一种计算机模拟出来的假斜体，即简单地对普通字母进行倾斜以模拟斜体风格。这两种方式的质量是有所不同的。

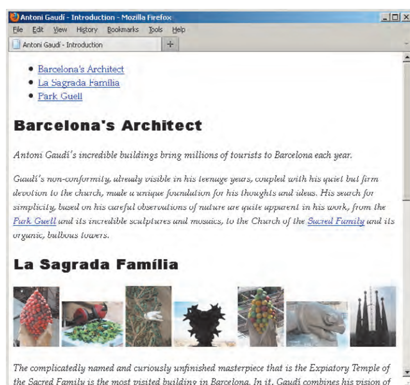


图 10.3.2　段落以斜体显示，但顶部的列表和标题没有以斜体显示

此外，字体设计师还可能专门为字体创建倾斜版本，它们通常是对普通字母进行倾斜，同时可能对字符间距等进行微调而产生的，与普通字母相比，字母本身的形状是相同的。要使用字体的倾斜版本，可以设置 font-style: oblique;，不过这样做并不常见。如果字体没有斜体版本或倾斜版本，就会以伪斜体显示。

1. 创建斜体

(1) 输入 font-style:。

(2) 在 :（冒号）后输入 italic（创建斜体文本）或 oblique（创建倾斜文本）。（99% 的情况下都会使用 italic。在所有情况下，很难察觉使用 oblique 与 italic 的差异。）

2. 取消斜体

输入 font-style: normal。

提 示　想取消斜体的一个可能的原因是，要在从父元素中继承了斜体格式的段落中对某些文字进行强调。关于继承的更多细节，参考 7.3 节。

提 示　font-style 属性是继承的。

10.4　应用粗体格式

粗体格式可能是让文本突出显示的最常见、最有效的方式。例如，浏览器通常默认为 h1 ~ h6 添加粗体格式。同斜体一样，可以为任何格式添加或取消粗体。样式表为粗体提供了很强的灵活性，可以控制粗细的相对值。不过，字体本身通常并未包含与相对值对应的不同粗细版本，因此不同相对值下显示的效果常常是一样的（如果有疑问，不妨直接设为 bold），如图 10.4.1 和图 10.4.2 所示。

```
body {
     font-family: "Palatino Linotype",
     ➝ Palatino, serif;
}

h1,
h2 {
     font-family: Arial, Helvetica,
     ➝ sans-serif;}

h2 {
     font-weight: normal;
}

em,
a:link,
a:hover {
     font-weight: bold;
}
```

图 10.4.1　浏览器会自动为标题（如 h1、h2）添加粗体格式。我为所有的 h2 元素应用了正常的粗细，取消了自动添加的粗体格式，你应该可以看出页面的差异。同时，我为 em 文本及新的链接、鼠标停留的链接添加了粗体格式（参见图 10.4.2）（注意，我将标题的 font-family 从 Arial Black 字体改成了 Arial，仅为本节举例用。参见倒数第二条提示。）

1. 应用粗体格式

(1) 输入 font-weight:。

(2) 输入 bold，让文本显示为具有平均加粗值的粗体。这一属性值适用于大多数情况。

或者输入 bolder（更粗）或 lighter（更细）以设置相对当前粗细的值。

或者输入 100 ~ 900 之间的 100 的倍数，其中 400 代表正常粗细，700 代表粗体。如果使用的是具有多种粗细版本的字体，这种方法就很有用。

2. 取消粗体格式

输入 font-weight: normal。



图 10.4.2　h1 标题是以粗体显示的，而 h2 则拥有常规的粗细。新的链接被突出显示了，而访问过的链接则没那么显眼

提示　由于各种字体定义粗细的方式并不相同，因此预定义值在不同字体上的表现可能不一致。预定义值用以表示某一给定字体下的相对值。

提示　如果字体的粗细少于九种，或者这些粗细集中于范围的一端，那么一些数值就可能对应于同样的粗细。

提示　鉴于上两条提示，添加粗体样式的通行做法是简单地写成 font-weight: bold，这种方法总是行之有效的。

10

提示 哪些元素是有可能需要取消粗体的呢？这包括任何自动添加了粗体格式的元素（应该能想到 strong、h1 ~ h6 和 b），以及继承了父元素粗体格式的元素（参见 7.3 节）。

提示 在本章接下来的例子中，h1 和 h2 标题的 font-family 属性仍设为 Arial Black，而不是图 10.4.1 ~ 图 10.4.3 中设置的 Arial。不过，在剩余的例子中，仍将为 h1 和 h2 标题保留 font-weight: normal; 这一设置，这是因为 Arial Black 本来就是一种加粗字体。如果为 Arial Black 设置 font-weight: bold;，浏览器会试着让它们变得更粗，显示为一种假粗体。为这些 Arial Black 字体的标题设置 font-weight: normal;，这些字体就会回到自然粗细状态。关于假斜体的讨论，参见 10.3 节。

提示 font-weight 属性是继承的。

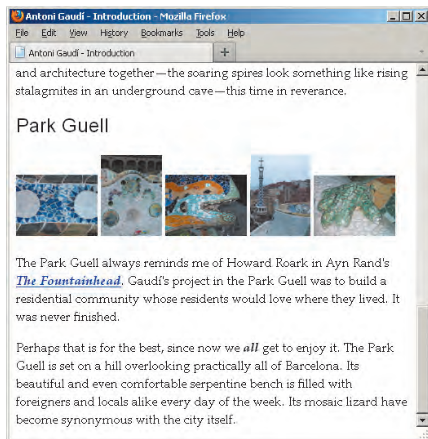


图 10.4.3　在页面的底部，可以看到一个链接（*The Fountainhead*）和单词 *all*。它们不仅根据新设的样式规则而显示为粗体，而且由于浏览器为包含它们的元素设置的默认样式而显示为斜体。它们分别由反映其含义的 cite 和 em 元素进行标记（同时应注意到，“Park Guell” h2 具有常规的粗细，它并不是以粗体显示的）

10.5　设置字体大小

为网页里的文本设置字体大小有两种方式：直接指定要使用的特定字号（如图 10.5.1 和图 10.5.2 所示），或指定相对于父元素文本的大小（如图 10.5.3 所示）。

```
body {
     font-family: "Palatino Linotype",
Palatino, serif;
     font-size: 14px;
}

h1,
h2 {
     font-family: "Arial Black", Arial,
     → sans-serif;
     font-weight: normal; /* 对已经加粗的
     → Arial Black取消假粗体 */
}

h1 {
     font-size: 22px;
}

h2 {
     font-size: 15px;
}

em,
a:link,
a:hover {
     font-weight: bold;
}

/* 目录导航 */
.toc a {
     font-size: 12px;
}
```

图 10.5.1　这里使用像素值控制文本的初始字体大小（我设置的字体大小都比大多数浏览器的默认值小一些）。段落继承了 body 里设置的 font-size。结果如图 10.5.2 所示

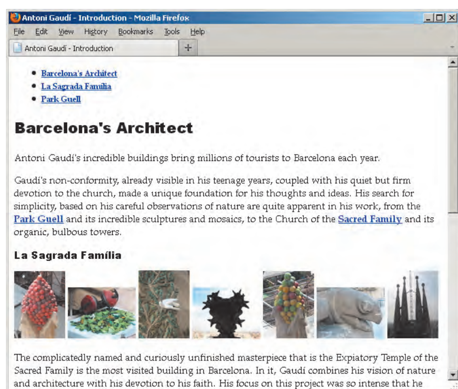


图 10.5.2 浏览器中显示的是上面指定的字体大小。目录（在页面顶端）中的链接、标题和段落都反映了样式表中设置的 font-size

设置相对于父元素的字体大小需要一点时间去习惯。你需要理解浏览器处理这些相对于父元素的单位的方式，下面花一点时间解释一下。

不过首先要说明，使用这种方法时，最好在 body 元素上建立一个基准，即声明 body { font-size: 100%; }（参见图 10.5.3）。大多数时候，这项设置等价于将字体大小设为 16px，即大多数系统的默认字体大小。照例，该属性值传递至其他的元素（记住，font-size 属性是继承的），除非该元素设置了它自己的 font-size。

那么，如何计算要指定的 em[①]值呢？实际上，1em 就等于默认的字号大小（在这里为 16px）。据此可以通过一点点除法确定 em 值（或百分比）。

要指定的字体大小 / 父元素的字体大小 = 值

例如，如果要将 h1 设为 22px，同时已知其父元素的字体大小为 16px，而

22 / 16 = 1.375

```
body {
    font-family: "Palatino Linotype",
    ➝ Palatino, serif;
    font-size: 100%; /* 16px */
}

h1,
h2 {
    font-family: "Arial Black", Arial,
    ➝ sans-serif;
    font-weight: normal;
}

h1 {
    font-size: 1.375em; /* 22px / 16px */
}

h2 {
    font-size: .9375em; /* 15px / 16px */
}

p {
    font-size: .875em; /* 14px / 16px */
}

em,
a:link,
a:hover {
    font-weight: bold;
}

/* 目录导航 */
.toc a {
    font-size: .75em; /* 12px / 16px */
}
```

图 10.5.3 body 里的 font-size: 100% 声明为 em 字体大小设置了参考的基准。这里的 100% 将被翻译为默认字体大小（大多数系统下为 16px）。这样，样式表的结果同图 10.5.1 所示的结果便是一样的。每个 font-size 属性值后面的注释解释了该值的计算方法，同时显示了等价的像素值

因此，设置 h1{font-size: **1.375em**;} 就可以了（参见图 10.5.3）。这条规则表示“将 h1 文本的大小设置为其父元素文本大小

10

① 在排版领域，em 是一种量度单位，1em 等于当前指定的字号大小。em 这一名称与字母 M 有关，起初，1em 等于给定字体环境下字母 M 的宽度。——译者注

的 1.375 倍”。另一种方式是写成 `h1 { font-size: 137.5%; }`。不过，除了在 body 中设置基准 font-size 时使用百分数以外，设置字体大小时使用 em 比百分数更为常见。

再如，要将段落文本设置为 14px，而

14 / 16 = 0.875

因此，设置 `p { font-size: .875em; }`（参见图 10.5.3）（这个值也可以设成 87.5%）。

再讨论一个例子。这可能是你感到困惑的地方。第一个段落包含两个链接（如图 10.5.4 和图 10.5.5 所示）。假设要让链接显示为 16px，同时让段落里的其他文本仍显示为 14px。你可能打算将链接的 font-size 设为 1em，因为 1em = 16px。

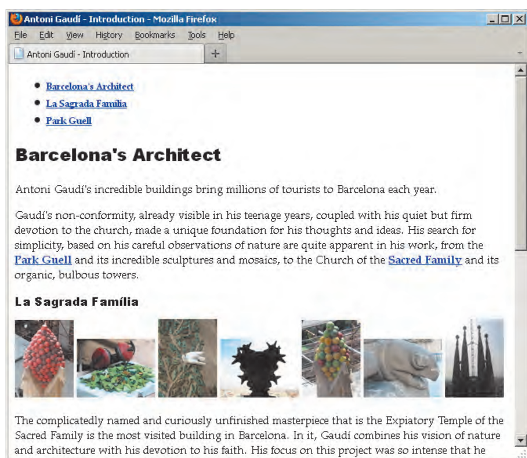


图 10.5.4　在大多数系统中，如果使用默认设置，以 em 为基础的字体大小同以像素为基础的版本是一样的（参见图 10.5.2）

```
...
<p>Gaudí's non-conformity, already visible in his teenage years, coupled with his quiet but firm
→ devotion to the church, made a unique foundation for his thoughts and ideas. His search for
→ simplicity, based on his careful observations of nature are quite apparent in his work, from the
→ <a href="#park-guell">Park Guell</a> and its incredible sculptures and mosaics, to the Church
→ of the <a href="#sagrada-familia">Sacred Family</a> and its organic, bulbous towers.</p>
...
```

图 10.5.5　这是 HTML 片段，其中有两个 a 元素，嵌套在父元素 p 中

需要记住的是，该值应该是相对于这些元素的父元素的。在这个例子中，它们的父元素是 p，而不是 body。段落的字体大小是 14px，因此，要让链接显示为 16px，需要设置一个比 1 大的 em 值。

16 / 14 = 1.1428457

因此，使用这个有点冗长的 `a { font-size: 1.1428457em; }` 将得到想要的结果。

最后需要指出的是，在大多数情况下，100% 的 body font-size 等于 16px。不过有一种例外的情况，即用户对浏览器的设置覆盖了该默认值。例如，如果他们是视障人士，他们可能将默认字体大小设置为 20px。将 body 设置为 100%，页面就会以此为基准设置其余的文本的相对大小。这是使用 em 和百分数设置字体大小的美之所在。

1. 指定特定字体大小

(1) 输入 font-size:。

(2) 在冒号（:）后输入准确的字号，如 13px。

或者使用关键词指定字体大小，即使用 xx-small、x-small、small、medium、large、x-large 或 xx-large。

提 示　关于单位的详细信息，参见 7.4 节。

提 示　数字与单位之间不应有任何空格。

提示 如果以像素为单位设置字体大小，使用 Internet Explorer 的访问者将无法使用浏览器的文本大小选项对文本进行放大或缩小。这是要用 em 或百分比控制字体大小的原因之一。从 IE7 开始，访问者可以对整个网页放大或缩小，这是对 IE6 的一项改进，因为这不仅是对字体大小进行改变。如果你想知道 IE6 的市场份额（它没有页面缩放功能），可以看到，在过去的几年里，它的份额已经大大地减少了，因此不少设计人员和开发人员已经完全不考虑它了（它还有不少臭名昭著的错误）。不过，它在某些国家仍然具有较大的用户群，尤其是中国和韩国。要查看它在世界各地的份额，可以访问 www.ie6countdown.com。

提示 不同的浏览器对关键词的解释方式可能有所不同。

提示 磅（pt）只能用做打印样式表的单位，不能用于屏幕样式表。

提示 由于屏幕分辨率有很大的差异，因此要避免在 font-size 中使用厘米、毫米和十二点活字[①]。实践中很少用到它们。

提示 font-size 属性是继承的。

2. 依父元素设置字体大小

(1) 输入 font-size:。

(2) 在冒号（:）后输入相对值，如 1.5em 或 150%。

或者使用相对关键词，即使用 larger 或 smaller（这种用法没有百分数常见，而百分数又没有 em 常见）。

提示 1 个 em 单位（不要与 HTML 的 em 元素弄混）等于所用字体的大小，因此，1em 等于 100%。

提示 父元素的字体大小可能是由用户或设计人员设置的，可能是继承的，也可能来自浏览器的默认设置。上文已经提到，大多数浏览器对 body 元素设置的默认字体大小为 16 像素。

提示 设置了相对字体大小的元素的子元素会继承这个大小，而不是继承相对值。因此，p（参见图 10.5.5）中 a 元素会继承 14 像素的字体大小（参见图 10.5.3），而不是相对值 0.875em。链接将显示为 14px，如果该样式没有被覆盖的话。

提示 字体大小可以同其他字体值一起进行设置。参见 10.7 节。

提示 CSS3 引入了一些新的单位，其中很有意思的一个便是 rem（root em 的简称）。它同 em 很像，不过它总是以根元素为参照系，因此不必像使用 em 时那样考虑父元素的字体大小。现代浏览器对它的支持程度很高。Internet Explorer 直到 IE9 才开始支持它（http://caniuse.com/#search=rem），因此需要为 IE 的早期版本提供默认值。Jonathan Snook 对如何使用 rem 进行了描述，并提供了为 IE8 及以下版本准备的解决方案（http://snook.ca/archives/html_and_css/font-size-with-rem）。（推荐将 body 字体大小设为 100%，而不是该作者所用的 62.5%，并据此创建其他的 rem 值。）

① 十二点活字是印刷的专用单位，1 十二点活字等于 1/6 英寸。——译者注

提 示 此外还有一个单位 ex，它指的是父元素的 x 高度，但浏览器对它的支持情况不好。

10.6　设置行高

行高指的是段落的行距，即段落内每行之间的距离。使用大一些的行高有时候会使主体文本更容易阅读（参见图 10.6.1 和图 10.6.2）。对于超过一行的标题，使用较小的行高则会让它们看起来更美观。

```
body {
     font-family: "Palatino Linotype",
     → Palatino, serif;
     font-size: 100%;
}

h1,
h2 {
     font-family: "Arial Black", Arial,
     → sans-serif;
     font-weight: normal;
}

h1 {
     font-size: 1.375em;
}

h2 {
     font-size: .9375em;
}

p {
     font-size: .875em; /* 16px / 14px */
     line-height: 1.6;
}

...
```

图 10.6.1　假设 body 元素默认大小为 16 像素，p 元素的字体大小为 0.875em，即大约 14 像素。行高将是 14 像素的 1.6 倍，即大约 22.4 像素

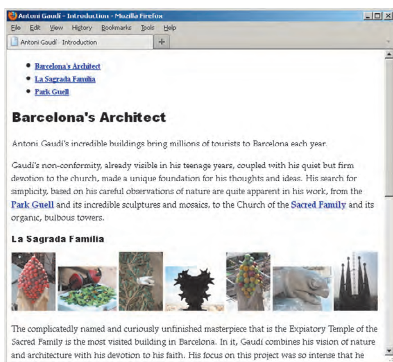


图 10.6.2　使用 line-height 拉大行距可以使它们更引人注目，并且更容易阅读

设置行高的步骤

(1) 输入 line-height:。

(2) 输入 *n*，这里的 *n* 是一个数字，它与元素的字体大小相乘，得出需要的行高。（这是最为常用的方法，即一个没有单位的数字。）

或者输入 *a*，这里的 *a* 是以 em、像素或磅（仅在打印样式表中使用磅）为单位的值。

或者输入*p*%,这里的*p*%是字体大小的百分数。

提 示 根据下一节将要讲到的，行高可以同字体系列、大小、粗细、字体样式和变体一起进行设置。

提 示 如果使用数字设定行高，那么所有的子元素都会继承这个因子。因此，如果父元素的字体大小是 16 像素（或以 em 等表示的等价大小），行高是 1.5，则该元素的行高就是 24（即 16×1.5）像素。如果子元素的字体大小是 10 像素，则该元素的行高就是 15（即 10×1.5）像素。

提 示 如果使用百分数或 em 值，那么只会继承产生的行高（即计算出来的值）。因此，如果父元素的字体大小是 16 像素，行高是 150%，则该元素的行高就是 24 像素。所有的子元素都将继承 24 像素的行高，不管字体大小是多少。

10.7 同时设置所有字体值

可以同时设置字体样式、粗细、变体、大小、行高和字体系列（如图 10.7.1 所示）。任何时候，只要可以，都应该采用这种方式，以保持样式表的简洁。

```
body {
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
}

h2 {
    font-size: .9375em;
}

p {
    /* 这些声明无法使用font简记法，除非
     → 同时声明字体系列 */
    font-size: .875em;
    line-height: 1.6;
}

em,
a:link,
a:hover {
    font-weight: bold;
}

/* Table of Contents navigation */
.toc a {
    font-size: .75em;
}
```

图 10.7.1　这个样式表与图 10.6.1 中的样式表是等效的，其效果如图 10.7.2 所示。这里只是将 body、h1 和 h2 样式规则的 font 属性合并了。注意，不必将 h1 和 h2 的 font-weight 设为 normal，因为 normal 是 font 属性的默认值。同时，这里无法对 p 元素的声明进行合并，因为 font 简记法至少要包含字体系列和字体大小的属性。关于包含了 font-style、font-variant、font-weight 和 line-height 的 font 简记法，参见第一条提示中的例子

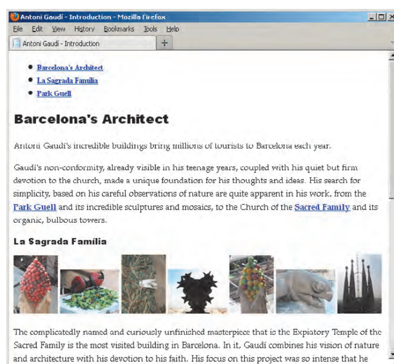


图 10.7.2　这个页面与图 10.6.2 是相同的

同时设置所有字体值的步骤

(1) 输入 font:。

(2) 可选步骤，输入 normal、italic 或 oblique 以设置字型（参见 10.3 节）。

(3) 可选步骤，输入 normal、bold、bolder、lighter 或 100 的倍数（最大到 900）以设置粗细（参见 10.4 节）。

(4) 可选步骤，输入 normal 或 small-caps 来取消或设置小型大写字母（参见 10.15 节）。

(5) 可选步骤，输入需要的字体大小（参见 10.5 节）。

(6) 如果需要，输入 */line-height*，这里的 *line-height* 是行与行之间的距离（参见 10.6 节）。

(7) 输入一个空格，再按优先次序输入需要的字体系列，以逗号分隔（参见 10.1 节）。

提示　结合了 font-size、line-height 和 font-family 声明的 font 简记法的一个例子是 font: .875em/1.6 "Palatino Linotype", Palatino, serif;。行高跟在字体大小和一个斜杠后面。还可以包含 font-style、font-variant 和 font-weight。下面是包含了所有可能属性的 font 声明：font: italic small-caps bold .875em/1.6 "Palatino Linotype", Palatino, serif;。属性的顺序很重要。可以使用这些属性的任意组合，只要其中声明了字体大小和字体系列。

10

提示　可以分别设置这些属性，但只要有可能，就应该将它们合并到 font 简记法里。

提示　头三个属性可以按任意顺序指定，也可以忽略。如果忽略它们，它们就被设为 normal（这也许不是你想要的），参见图 10.7.1。

提示　字体大小和字体系列属性必须始终显式地声明：先是字体大小，再是字体系列。

提示　行高是可选的，但它如果出现，就必须紧跟在字体大小和斜杠后面。

提示　font 属性是继承的。

10.8 设置颜色

可以修改网页里元素的颜色，如图 10.8.1 和图 10.8.2 所示。

```
body {
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
}

h2 {
    font-size: .9375em;
}

p {
    font-size: .875em;
    line-height: 1.6;
}

em {
    font-weight: bold;
}

/* 链接 */
a:link {
    color: #74269d;
    font-weight: bold;
}

a:visited {
    color: #909;
}

a:hover {
    color: #c3f;
    font-weight: bold;
}

/* 目录导航 */
.toc a {
    font-size: .75em;
}
```

图 10.8.1　可以使用颜色名、十六进制数或 RGB、HSL、RGBA、HSLA 值定义颜色。注意，a:visited 和 a:hover 颜色（分别为 #909 和 #c3f）使用了第二条提示里讲到的简称



图 10.8.2　标题是深蓝色的，文本是浅紫色的。链接是深紫色的，但在访问之后会变浅，当鼠标停留时又会变成粉紫色以突出显示（另见彩插）

设置颜色的步骤

(1) 输入 color:。

(2) 输入 *colorname*，这里的 *colorname* 是预定义颜色中的一种（参见 7.4 节中的“CSS 颜色”）。

或者输入 *#rrggbb*，这里的 *rrggbb* 是颜色的十六进制呈现。这是最常用的指定颜色的方法。

或者输入 rgb(*r*, *g*, *b*)，其中 *r*、*g*、*b* 是 0 ~ 255 之间的整数，分别表示所需颜色里红、绿、蓝的量。

或者输入 rgb(*r*%, *g*%, *b*%)，其中 *r*、*g*、*b* 分别是所需颜色里红、绿、蓝的百分数。

或者输入 hsl(*h*, *s*, *l*)，其中 *h* 是 0 ~ 360 之间的数值，表示所需颜色的色相；*s* 和 *l* 均是百分数，分别表示所需颜色的饱和度和亮度。（一般来说，对于不透明颜色，使用十六进制数或 RGB 值更好。）

或者输入 rgba(*r*, *g*, *b*, *a*)，其中 *r*、*g*、*b* 是 0 ~ 255 之间的整数，分别表示所需颜色里红、绿、蓝的量；*a* 是 0 ~ 1 之间的小数，表示所需颜色的 alpha 透明度。

或者输入 hsla(*h*, *s*, *l*, *a*)，其中 *h* 是 0 ~ 360 之间的数值，表示所需颜色的色相；*s* 和 *l* 均是百分数，分别表示所需颜色的饱和度和亮度；*a* 是 0 ~ 1 之间的小数，表示所需颜色的 alpha 透明度。

提示 color 属性是继承的。

提示 如果输入的 r、g 或 b 的值大于 255，就会使用 255。类似地，高于 100% 的百分数将被替换为 100%。

提示 当十六进制数值由重复的数字组成时，还可以使用 #rgb 设置颜色。事实上，这是推荐的做法。因此，可以（且应该）将 #FF0099 写做 #F09 或 #f09。

提示 十六进制数不应该用双引号包围。

提示 记住，Internet Explorer 在 IE9 之前的版本不支持 HSL、RGBA 和 HSLA，因此，如果要在颜色声明中使用这些记法，就需要为 IE 的旧版本定义备用颜色。详细说明参见 7.4 节中的“CSS 颜色”。

10.9 修改文本的背景

可以为单个元素设置背景，或者为整个页面设置背景，还可以为上述二者的任意组合设置背景（参见图 10.9.1 和图 10.9.2）。如此，便可以对几个段落、几个单词、不同状态的链接、内容区域等修改背景。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    ➝ serif;
}

… [其他CSS] …

/* 目录导航 */
.toc {
    background: #ebc6f9;
}

.toc a {
    font-size: .75em;
}
```

图 10.9.1 对 body 元素设置背景颜色就是为整个页面设置背景颜色。对属于 toc 类的元素设置背景可以让目录与页面其他内容区分开来（参见图 10.9.2）

修改文本背景的步骤

(1) 输入 background:。

(2) 输入 transparent（透明）或 *color*，这里的 *color* 是颜色名称、十六进制数值，

或 RGB、HSL、RGBA、HSLA 颜色值（参见 10.8 节）。十六进制值颜色是最为常用的。

(3) 如果愿意，输入 url(*image.gif*) 以使用图像作为背景，这里的 *image.gif* 是图像相对于样式表所在位置的路径和文件名。

如果愿意，输入 repeat 将图像在横向和纵向重复。输入 repeat-x 仅横向重复，或输入 repeat-y 仅纵向重复，或者 no-repeat 不重复。

如果愿意，输入 fixed（固定）或 scroll（滚动）以决定背景图是否应该随画布一起滚动（通常不设置此项；如果不设置，则使用默认值 scroll）。

如果愿意，输入 *x y* 以设置背景图像的位置，其中 *x* 和 *y* 可以表示为距离左上角的绝对值或百分数。或者用 left（左对齐）、center（居中）或 right（右对齐）表示 *x*，用 top（顶端对齐）、center（居中）或 bottom（底端对齐）表示 *y*。

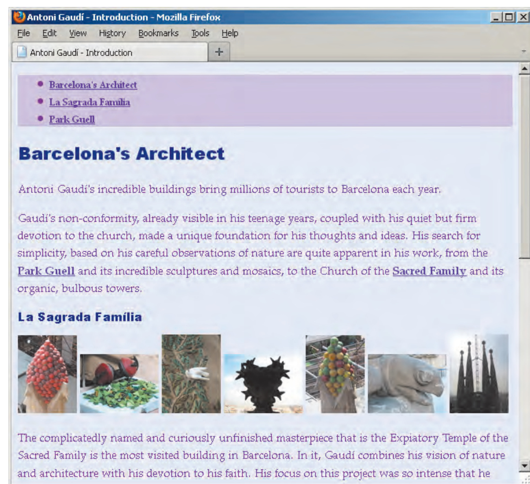


图 10.9.2　body 元素的背景是浅蓝色的，目录的背景则是浅紫色的（另见彩插）

提示　可以同时指定背景颜色和背景图像的 URL。在图像尚未加载前，或出于某种原因未能加载时，就会使用这个颜色，并且会在图像的透明部分露出这个颜色。如果为元素定义了背景图像，通常一种好的做法是同时为其定义背景颜色，这种颜色要让文字颜色和背景具有足够的对比度。这样做就能确保在背景图像被用户关闭或因为某种原因未能加载时文字依然具有可读性。如果不显式地定义背景颜色，该元素就会继承其父元素的颜色，如 body 元素默认的白色背景。如果你想在深色背景图像上显示浅色文字，就有可能产生问题。更多细节参见“更多关于背景的说明”。

提示　背景和前景之间应有足够的对比度，从而让访问者能够轻松地阅读文字。这样做不仅能帮助普通用户，而且对可访问性来说也是很重要的。对比度对色盲访问者来说尤其重要。

提示　background 属性不是继承的。

多重背景及更多的 CSS3 特性

CSS3 引入了几个新的背景相关的特性，包括期待已久的多重背景、背景缩放等。其中的一些可以在第 14 章学到。如果要深入了解相关 CSS3 模块中的所有这些特征，参见 www.w3.org/TR/css3-background/。

更多关于背景的说明

background 属性非常强大，你会发现很多需要用上它的场合。容易想到的是你可能对本节中的第 (3) 步仍有些许疑惑。

下面是一个例子：

```
body {
    background: #foc url(bg-page.png) repeat-x scroll 0 0;
}
```

这实际上是一种简记法，就像使用 font 属性将 font-family、font-size、line-height 等属性组合在一条声明里一样。

从左至右对 background 简记法进行分解，可以重新写为：

```
body {
    background-color: #foc;
    background-image: url(bg-page.png);
    background-repeat: repeat-x;
    background-attachment: scroll;
    background-position: 0 0;
}
```

这样做代码量就变大了，因此，很容易理解为什么应该使用简记法，除非有理由分开书写。实际上，甚至还可以将 scroll 和 0 0 去掉，让示例变得更短一些。

```
body {
    background: #foc url(bg-page.png) repeat-x;
}
```

实践中，URL 可能是类似于 ../img/bg-page.png 的形式，因为我们通常不希望将图像与样式表保存在同一目录。

那么，这代表什么呢？设想背景图像 bg-page.png 是一个 15 像素宽、600 像素高的重复图案。示例样式规则表示：“在水平方向无限重复这个图像，在任何没有图像的地方无限使用 #foc 颜色。”这样，就会看见图像在纵向前 600 像素的区域平铺开来。如果内容的高度大于 600 像素，就会看见 #foc 颜色（粉红色，很适合用于 Hello Kitty 礼品站）。

背景图像应该很好地与背景颜色融合，以免访问者在图像与颜色交接处看到一条明显的线。由于 background 是为 body 定义的，因此所有的页面内容都覆盖在背景图像与背景颜色上。如果你研究一下其他网站的 CSS，很容易发现为 body 设置的 background 的一些变化。

下面给出了更多的例子，介绍了一些可能的处理方式。

黑色的背景颜色，结合在纵向上无限重复显示的图像。

```
body {
    background: #000 url(../../image/bg-page.png) repeat-y;
}
```

在所有方向无限重复显示的背景图像。在图像未能加载时，或在图像加载之前，显示黄色。

```
body {
   background: yellow url(../img/bg-smiley-faces.png);
}
```

深绿色的背景颜色，结合不重复显示且定位于距页面左边缘 200 像素、距上边缘 125 像素的图像。允许使用负数。使用 center 可以让它相对页面居中。

```
body {
   background: #3f8916 url(../../img/bg-gumby.png) no-repeat 200px 125px;
}
```

上面的介绍集中于 body 背景，这是因为它们对页面的设计有着极大的影响。实际上，可以将 background 属性应用于任何元素。因此，如果你愿意，完全可以将 Telly Savalas 的照片设为所有段落的背景。我**鼓励**你这样做。

10.10　控制间距

可以增加或减少单词之间或字母之间的距离（参见图 10.10.1 和图 10.10.2）。前者称为字间距（tracking），后者称为字偶距（kerning）。

```
body {
     background: #eef;
     color: #909;
     font: 100% "Palatino Linotype", Palatino,
     → serif;
}

h1,
h2 {
     color: navy;
     font: 1.375em "Arial Black", Arial,
     → sans-serif;
     letter-spacing: .4em;
}

h2 {
     font-size: .9375em;
}

... [其余的CSS] ...
```

图 10.10.1　这里为标题字母添加了 0.4em 的额外间距。在字体大小为 22 像素的情况下，这意味着单词之间的距离几乎有 9 像素（参见图 10.10.2）

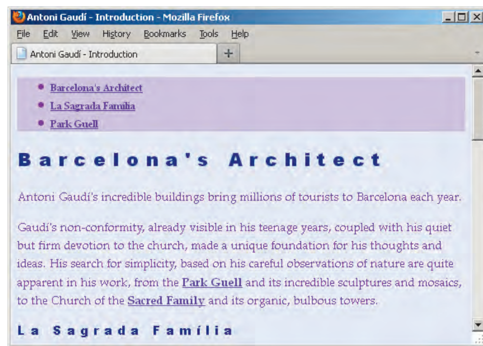


图 10.10.2　现在，标题中字母之间的距离变大了

1. 指定字间距

输入 word-spacing: *length*，这里的 *length* 是一个带单位的数字，如 0.4em 或 5px。

2. 指定字偶距

输入 letter-spacing: *length*，这里的 *length* 是一个带单位的数字，如 0.4em 或 5px。

提 示　可以对单词间距和字母间距使用负数。

提示　单词间距和字母间距的值还可能受到所选的对齐方式和字体系列的影响。

提示　要将字母间距和单词间距设为默认值（即不添加额外的间距），可以使用 normal 或 0。

提示　如果要使用 em 值，那么只有产生的大小（即“计算出来的值”）会被继承。因此，如果父元素字体大小为 16 像素，额外的单词间距为 0.1em，则每个单词之间的额外间距为 1.6 像素。同时，所有子元素每个单词之间也有 1.6 像素的额外间距，不管它们的字体大小是多少。如果要覆盖继承的值，可以显式地为子元素设置间距。

提示　word-spacing 和 letter-spacing 属性是继承的。

10.11　增加缩进

通过设置 text-indent 属性，可以指定段落第一行前面应该空出多大的空间，如图 10.11.1 和图 10.11.2 所示。

增加缩进的方法

输入 text-indent: *length*，这里的 *length* 是一个带单位的数字，如 1.5em 或 18px。

提示　使用负数会产生悬挂缩进。使用悬挂缩进时，可能还需要增加文字框周围的内边距或外边距，从而让容器可容纳伸到外边的文本。（参见 11.8 节和 11.9 节。）

提示　与通常情况一样，em 值要根据元素的字体大小进行计算，百分数要根据父元素的宽度进行计算。

提示　text-indent 属性是继承的。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
    letter-spacing: .4em;
}

h2 {
    font-size: .9375em;
}

p {
    font-size: .875em;
    line-height: 1.6;
    text-indent: 1.5em;
}
... [其余的CSS] ...
```

图 10.11.1　这段代码为 p 元素添加了 1.5em 的缩进。由于字体大小约为 14 像素，因此这个缩进约为 21 像素（参见图 10.11.2）

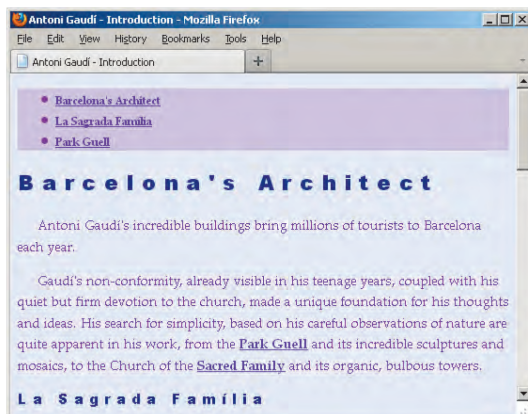


图 10.11.2　每个段落都有 21 像素的缩进

10

提示　如果要使用百分数或 em 值，那么只有生成的大小(即“计算出来的值”)会被继承。因此，如果父元素为 300 像素宽，那么 10% 的 text-indent 就是 30 像素，同时，所有子元素第一行也都会缩进 30 像素，而不管它们各自父元素的宽度是多少。

提示　要去除继承的缩进，使用 0 即可。

10.12　设置空白属性

默认情况下，HTML 文档里的多个空格和回车会显示为一个空格，或者被忽略。如果要让浏览器显示这些额外的空格，可以使用 white-space 属性。

设置空白属性的步骤

(1) 输入 white-space:;

(2) 输入 pre，以让浏览器显示原文本中所有的空格和回车；

或者输入 nowrap，以使所有空格成为非断行空格；

或者输入 normal，以按正常方式处理空格。

图 10.12.1 和图 10.12.2 演示了如何对第一个段落应用 nowrap 值。可以看到，这样做，该段落就不会自动换行了（参见图 10.12.3）。不过，我们并非真的希望为页面添加这种效果，因此在接下来的例子中，这个 class 及相关的 CSS 将被移除。

提示　white-space 属性的 pre 值的名称来源于 pre 元素，该元素以等宽字体显示其包含的文本，并保留所有的空格和回车。pre 元素的名称来自单词 pre-formatted（预格式化）。关于 pre 的更多信息，参见第 4 章。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
    letter-spacing: .4em;
}

h2 {
    font-size: .9375em;
}

p {
    font-size: .875em;
    line-height: 1.6;
    text-indent: 1.5em;
}

.intro {
    white-space: nowrap;
}
... [其余的CSS] ...
```

图 10.12.1　将 white-space 的值设为 nowrap，空格将作为非断行空格进行处理

```
...

<h1 id="gaudi">Barcelona's Architect</h1>

    <p class="intro">Antoni Gaudi's
    → incredible buildings bring millions of
    → tourists to Barcelona each year.</p>

    <p>Gaudi's non-conformity, already
    → visible in his teenage years, coupled
    → with his quiet but firm devotion to
    → the church, made a unique foundation
    → for his thoughts and ideas...</p>

...
```

图 10.12.2　仅仅出于显示目的，为第一个段落添加了 intro 类，从而可以看到 nowrap 对第一段显示的影响



图 10.12.3　第一个段落不会换行，即便浏览器窗口太窄，不足以显示整行，这样就会出现横向滚动条

提 示　注意，将 white-space 属性的值设为 pre 不会影响元素的字体（这与 pre 元素不一样，浏览器默认会以等宽字体显示其包含的文本）。

提 示　在具有 white-space: nowrap 样式的元素中，可以使用 br 元素手动创建换行。尽管如此，还是要尽量避免使用 br，除非没有更好的选择，因为这样做会在 HTML 中混合表现，而不是让 CSS 控制它。关于 br 元素的更多信息参见 4.16 节。

10.13　对齐文本

根据需要，可以让文本左对齐、右对齐、居中对齐或两端对齐（参见图 10.13.1 和图 10.13.2）。

对齐文本的步骤

(1) 输入 text-align:;

(2) 输入 left 让文本左对齐；

或者输入 right 让文本右对齐；

或者输入 center 让文本居于屏幕的中间；

或者输入 justify 让文本两端对齐。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
    letter-spacing: .4em;
    text-align: center;
}

h2 {
    font-size: .9375em;
}

p {
    font-size: .875em;
    line-height: 1.6;
    text-align: justify;
    text-indent: 1.5em;
}

... [其余的CSS] ...
```

图 10.13.1　做出以下更改以后，标题和段落文本的对齐方式就会相应调整。不要忘记 text-align 中的连字符（-）

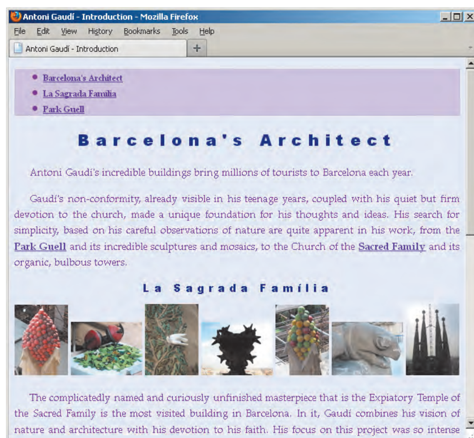


图 10.13.2　做出更改以后，标题变成居中对齐，段落文本变成两端对齐

提示　如果选择让文本两端对齐，要注意单词间距和字母间距有可能受到严重的影响。更多信息参见 10.10 节。

提示　注意，text-align 属性只能用于设为 display: block 或 display: inline-block 的元素。p 和 div 等元素已默认设为 display: block。在 HTML5 之前，这类元素称为块级元素。它们的默认设置在 HTML5 中被保留了，但它们已不再称为块级元素了，以免将 HTML 语义与外观等同。因此这一点主要适用于短语内容（在 HTML5 之前称为“行内”元素），如 strong、em、a、cite 等出现在句子、标题上下文中的元素。如果要对这些元素单独设置对齐方式，而不是与包围它们的文本一同设置，必须先覆盖它们默认的 display: inline 样式，设置 display: block 或 display: inline-block，再设置相应的 text-align。对于 display: inline-block 的元素，可能需要设置一个宽度才能看见效果。事实上，需要为“行内”元素设置 text-align 的情形是非常少见的。

提示　text-align 属性是继承的。它的默认值取决于文档的语言和书写系统，不过在大多数情况下，它会被不加区分地设置为左对齐。

10.14　修改文本的大小写

使用 text-transform 属性，可以为样式定义文本的大小写（参见图 10.14.1）。通过这种方法，可以将文本显示为首字母大写、全部大写（参见图 10.14.2）、全部小写或按原样显示。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
    letter-spacing: .4em;
    text-align: center;
}

h1 {
    text-transform: uppercase;
}

h2 {
    font-size: .9375em;
}

... [其余的CSS] ...
```

图 10.14.1　将一级标题显示为全部大写，以便突出它（代码实现参见图 10.14.2）



图 10.14.2　现在，标题真的很醒目

修改文本大小写的步骤

(1) 输入 text-transform:。

(2) 在冒号（:）后输入 capitalize 让每个单词的首字母大写；

或者输入 uppercase 让所有字母大写；

或者输入 lowercase 让所有字母小写；

或者输入none让文本保持本来的样子(可以用来取消继承的值)。

提示 capitalize属性值有它的局限性。它并不了解一门语言里按照惯例首字母不应该大写的单词，它只是简单地让每个单词首字母大写。因此，HTML中的文本Jim Rice enters the Hall of Fame将显示为Jim Rice Enters The Hall Of Fame。

提示 既然可以改变HTML里的文本，为什么还要用text-transform呢？这是因为，有时，内容是你无法控制的。例如，内容可能存储在数据库里，或者来自另一个网站的新闻源。在这些情况下，只能通过CSS控制文本的大小写。此外，如果要让文本全部大写，大多数时候应使用text-transform: uppercase。搜索引擎通常是按它在HTML里输入的样子索引的，在搜索结果里显示标准的大小写会更容易阅读。

提示 lowercase属性值可以用来创建特殊的标题样式（就像诗人e.e. cummings[①]）。

提示 text-transform属性是继承的。

10.15 使用小型大写字母

很多字体都有对应的小型大写字母变体，其中一些字母是大写的，但缩小到了小写字母的大小。可以使用font-variant调用小型大写字母变体（参见图10.15.1和图10.15.2）。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
    letter-spacing: .4em;
    text-align: center;
}

h1 {
    text-transform: uppercase;
}

h2 {
    font-size: 1.15em;
    font-variant: small-caps;
}

... [其余的CSS] ...
```

10.15.1 这里为h2设置了small-caps，并将其字体大小增加了一些，从而让它与h1在比例上更加协调（参见图10.15.2）。不要忘记font-variant和small-caps中的连字符（-）



图10.15.2 现在可以看到h2中每个字母的小型大写版本。小型大写字母在不同浏览器中的显示效果稍有不同

① 美国诗人E. E. Cummings的一些作品全部使用小写字母，署名也写做e.e. cummings。——译者注

1. 使用小型大写字母的方法

输入 font-variant: small-caps。

2. 取消小型大写字母的方法

输入 font-variant: none。

提 示　与简单地缩小字号的大写字母相比，小型大写字母显得更为轻巧。

提 示　并非所有的字体都有对应的小型大写字母设计。如果浏览器没有找到这种设计，那么它有几种选择：可以简单地缩小大写字母的尺寸来模拟小型大写字母（这会使它们显得有点矮胖），也可以完全忽略小型大写字母，并将它们显示为全部大写（就像前面描述的 text-transform: uppercase 那样），理论上还可以选择字体列表中的下一个字体，看它有没有小型大写字母设计（现实中我从未见过这样处理的）。

提 示　font-variant 属性是继承的。

10.16　装饰文本

可以使用样式表对文本进行装饰，如添加下划线和删除线（可能用来表示修改），参见图 10.16.1 和图 10.16.2。

```
body {
    background: #eef;
    color: #909;
    font: 100% "Palatino Linotype", Palatino,
    → serif;
}

h1,
h2 {
    color: navy;
    font: 1.375em "Arial Black", Arial,
    → sans-serif;
    letter-spacing: .4em;
```

（续左栏代码）

```
    text-align: center;
}

h1 {
    text-transform: uppercase;
}

h2 {
    font-size: 1.15em;
    font-variant: small-caps;
}

p {
    font-size: .875em;
    line-height: 1.6;
    text-align: justify;
    text-indent: 1.5em;
}

em {
    font-weight: bold;
}

/* 链接 */
a:link {
    color: #74269d;
    font-weight: bold;
    text-decoration: none;
}

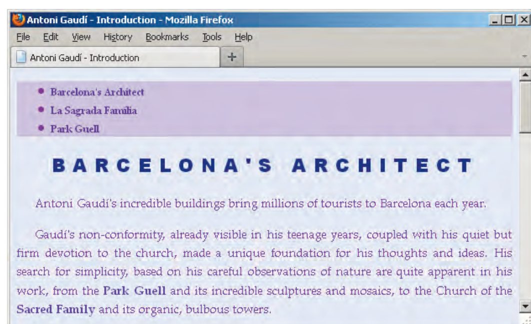
a:visited {
    color: #909;
    text-decoration: none;
}

a:hover {
    color: #c3f;
    font-weight: bold;
    text-decoration: underline;
}

/* 目录导航 */
.toc {
    background: #ebc6f9;
}

.toc a {
    font-size: .75em;
}
```

图 10.16.1　这是图 10.16.2 显示的整个页面的样式表，包括改变链接样式的 text-decoration。不过，下划线等文本装饰并不限于 a 元素，它们还可以应用于其他元素



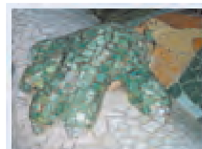




图 10.16.2 在最上面的图中，可以看到所有链接（包括目录中的链接）的下划线都被移除了。页面下方有另外一个链接，它以斜体显示，它是一本书的书名，因此笔者使用了位于链接里的 cite 元素标记（cite 元素的默认样式为斜体）。最下面的图显示了鼠标停留时链接的下划线样式，它们提醒访问者可以进一步采取行动。本章前面部分已经将 a:hover 的颜色设为 #c3f

1. 装饰文本的步骤

(1) 输入 text-decoration:。

(2) 在冒号（:）后输入 underline 以添加下划线；

或者输入 overline 以添加上划线；

或者输入 line-through 以添加删除线。

2. 取消文本装饰的方法

输入 text-decoration: none;。

提示 对于正常情况下有装饰的元素（如 a、del、ins）以及从父元素继承了装饰样式的元素，可以取消它们的装饰。

提示 将链接的下划线去掉固然很好，但是必须用别的方式将链接与周围的内容进行区分，否则访问者就无法知道它们是可激活的链接。

10

第 11 章

用 CSS 进行布局

11

本章内容

- ❑ 开始布局注意事项
- ❑ 建立页面结构
- ❑ 在旧版浏览器上为 HTML5 元素添加样式
- ❑ 对默认样式进行重置或标准化
- ❑ 盒模型
- ❑ 修改背景
- ❑ 设置元素的高度和宽度
- ❑ 设置元素周围的外边距
- ❑ 在元素周围添加内边距
- ❑ 使元素浮动
- ❑ 控制元素浮动的位置
- ❑ 设置边框
- ❑ 偏移自然流中的元素
- ❑ 对元素进行绝对定位
- ❑ 指定元素的三维位置
- ❑ 决定溢出的位置
- ❑ 垂直对齐元素
- ❑ 修改鼠标指针
- ❑ 显示和隐藏元素

可以使用 CSS 创建各种各样的布局。本章将演示如何创建一种常见的布局：顶部有一个报头，中间是两栏内容，底部有一个页脚，如图 11.0.1 所示。不过，使用即将讲到的 CSS 属性，还可以创建截然不同的布局。



图 11.0.1　这个页面包含两个流式栏、一个页眉和一个页脚，它是用 CSS 进行布局的。本章将逐步讲解这个页面的布局（另见彩插）

在本章中，不会将 CSS 的每一行都显示出来。例如，大部分文本格式化都在前面完成了。完整的代码见 www.bruceontheloose.com/htmlcss/examples/chapter-11/finished-page.html。我还创建了一个定宽（非流式）版本（文件名为 finished-page-fixed-width.html），以演示如何实现定宽布局。所有这些文件（尤其是样式表）中都包含了大量的注释，用以对代码进行解释。

11.1　开始布局注意事项

下面列举的一些要点有助于对网站进行

布局，并在发布之前对网站进行调整。

1. 内容与显示分离

- 作为最佳实践，应始终保持内容（HTML）与显示（CSS）分离。第 8 章介绍了如何通过外部样式表实现这一点。如果对所有的页面都这样做，就可以共享相同的布局和整体样式。这也让日后修改整个网站的设计变得更加容易——只修改 CSS 文件就可以了。

2. 浏览器注意事项

- 并非所有的访问者都使用同样的浏览器，同样的操作系统，甚至同样的设备访问你的网站。因此，大多数情况下，在将网站放到服务器上发布之前，通常需要在很多浏览器上对页面进行测试。推荐在开发过程中就用几个浏览器对页面定期进行测试，这样，在最后进行全面测试时，碰到的问题就会少一些。关于如何对页面进行测试，以及测试用浏览器的有关信息，参见 20.6 节。
- 有时，有必要针对 IE 的特定版本编写 CSS 样式规则，以修复 IE 的异常行为引起的问题。这对 IE6 和有所改善的 IE7 来说尤其常见。

  有几种办法可以实现上述要求，不过从性能上说，最好的方法是使用条件注释在 `html` 元素上创建 IE 版本特有的类，并在样式表中应用这个类。更多细节参见 http://paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/。这个页面包含的内容较多，如果你想看该用什么代码，可以直接转至标题“Throw it on the html tag”（同时请阅读代码后面的注解）。另一种方法是使用条件注释引入位于单独样式表中的 IE 补丁。

  我在本书网站的代码示例中提供了这两种技术的例子。第一种方法位于 finished-page.html（参见本章导语），第二种方法位于同一目录下的 finished-page-conditional-stylesheets.html。关于条件注释，参见 www.quirksmode.org/css/condcom.html。

3. 布局方法

有几种布局的方式。

- 对于固定（fixed）布局，整个页面和每一栏都有基于像素的宽度。顾名思义，无论是使用移动电话和平板电脑等较小的设备查看页面，还是使用桌面浏览器并对窗口进行缩小，它的宽度都不会改变。你在浏览万维网时已经见过不少固定布局的网站了，尤其是公司网站和大牌网站。固定布局也是学习 CSS 时最容易掌握的布局方式。
- 流式（fluid 或 liquid）布局使用百分数定义宽度，允许页面随显示环境的改变进行放大或缩小。这种方法后来被用于创建响应式（responsive）布局和自适应（adaptive）布局，这些布局方式不仅可以像传统的流式布局那样在手机和平板电脑上缩小显示，还可以根据屏幕尺寸以特定方式调整其设计。这就可以在使用相同HTML的情况下，为移动用户、平板电脑用户和桌面用户定制单独的体验，而不是提供三个独立的网站。（Ethan Marcotte 创建了术语“响应式 Web 设计”（responsive Web design）及其背后的技术包。想要对此有所了解，参见他发表在 *A List Apart* 上的文章（www.alistapart.com/articles/responsive-web-design/）。他在

Responsive Web Design 一书中对此作了深入探讨，强烈推荐读者去读一读。自适应布局使用了一些相同的技术。）

- 弹性（elastic）布局对宽度和其他所有属性的大小值都使用 em，从而让页面根据用户的 `font-size` 设置进行缩放。

没有一种布局方式可以适用于所有的情景，事实上还有一些混合的方式。本章将讲解如何创建一个混合流式布局和固定布局的布局：每栏都是流式的、基于百分数的宽度，从而可对其进行放大或缩小，不过整个页面宽度拥有一个固定的最大宽度，限制了它能放大到的宽度。

11.2　建立页面结构

使用 CSS 的主要原因是为了将格式及样式规则与页面的内容分离。这会让页面更容易维护，并为应对不同的浏览器、平台、设备，甚至打印的需求提供了灵活性。就像处理文本样式一样，CSS 也为页面的整体布局提供了各种方法。可以将 CSS 应用于第 3 章讲到的表现页面主体结构元素的内容容器，参见图 11.2.1 和图 11.2.2。使用 CSS，页面的报头、主内容区、侧栏、页面级页脚等都会使页面在视觉上变得生动起来。

```
...
<body>
<div id="container">
    <div id="page">
       <!-- ==== 开始报头 ==== -->
       <header id="masthead" role="banner">
          <p class="logo "><a href="/">photobarcelona&hellip; <span>capturing barcelona's cultural
          ➝ treasures on film</span></a></p>

          <div>
             <nav role="navigation">
                ... [链接列表] ...
             </nav>

             <form method="get" role="search">
                ...
             </form>
          </div>
       </header>
       <!-- 结束 #masthead -->

       <!-- ==== 开始主体内容 ==== -->
       <div id="main" role="main">
          <h1>Recent Entries</h1>
          <!-- 开始第1个条目 -->
          <section class="entry">
             <header>
                <h2 lang="es">Hospital Sant Pau</h2>
                <p class="date"><time datetime="2011-06-26" pubdate="pubdate">June 26, 2011</time></p>
             </header>

             ... [图像] ...

             <div class="intro">
```

（接下页未完代码）

（续上页未完代码）

```
            <p>The Saint Paul Hospital at the top ...</p>

            <p class="continued"><a href="#">continued</a></p>
         </div>
      </section>
      <!-- 结束 .entry #1 -->

      <!-- 开始第2个条目 -->
      <section class="entry">
         ...
      </section>
      <!-- 结束 .entry #2 -->

      <!-- 开始第3个条目 -->
      <section class="entry">
         ...
      </section>
      <!-- 结束 .entry #3 -->
   </div>
   <!-- 结束 #main 内容 -->

   <!-- ==== 开始侧栏 ==== -->
   <div id="related" class="sidebar" role="complementary">
      <aside class="excerpt">
         <h2>From my Window</h2>

         ...
      </aside>

      <aside class="archive">
         <nav role="navigation">
         <h2>Archive</h2>
            ... [链接列表] ...

            ...
         </nav>
      </aside>
   </div>
   <!-- 结束 #sidebar -->

   <!-- ==== 开始页脚 ==== -->
   <footer id="footer" role="contentinfo">
      <h1>about this photoblog</h1>

      ...

      ... [图像列表] ...
   </footer>
   <!-- 结束 #footer -->
  </div>
  <!-- 结束 #page -->
</div>
```

（续上页未完代码）

```
<!-- #container -->
</body>
</html>
```

图 11.2.1　这是贯穿本章的文档，四个主要区块（masthead、main、sidebar 和 footer）包含在两个外部包装容器（container 和 page）里。读者可以在本书网站上找到完整的文件（www.bruceontheloose.com/htmlcss/examples/chapter-11/finished-page.html）。默认情况下，页面很普通，但具有各种功能（参见图 11.2.2）

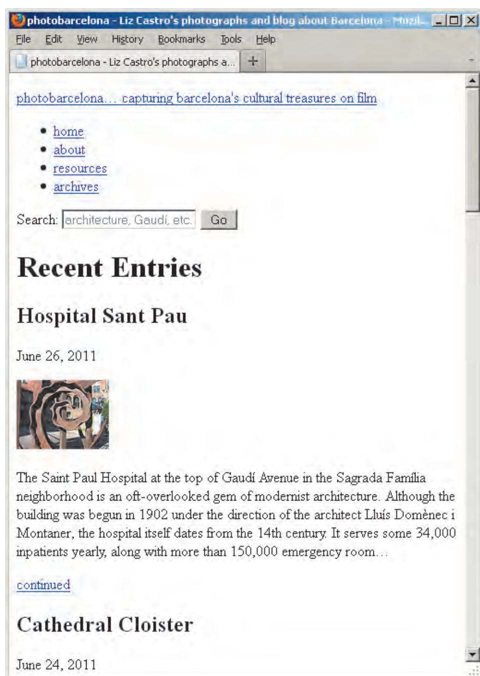


图 11.2.2　这是示例在浏览器里显示的样子，没有除浏览器默认样式以外的样式。由于它的语义结构非常好，页面是完全可用且可理解的，只是有一点朴素

建立页面结构的步骤

(1) 恰当地使用 article、aside、nav、section、header、footer 和 div 等元素将页面划分成不同的逻辑区块。根据需要，对它们应用 ARIA 地标角色。上述两点的详细说明见第 3 章。图 11.2.1 包括以下内容。

- 用于为页面应用一些设计并将页面包起来的容器和页面 div。
- 用作报头的 header，包括标识、标语、搜索框和主导航。
- 包含主体内容的主 div，同时将其划分为多个入口 section。
- 容纳月度观点和链接的侧栏 div。
- 包含“关于”等内容的页面级 footer 元素。

(2) 按照一定的顺序放置内容，确保页面在不使用 CSS 的情况下也是合理的（参见图 11.2.2）。例如，首先是报头，接着是主体内容，接着是一个或多个侧栏，最后是页面级的页脚。将最重要的内容放在最上面，对于智能手机和平板电脑等小屏幕用户（以及使用不支持 CSS 的老式移动设备的用户）来说，不用滚动太远就能获取主体内容。此外，搜索引擎“看到”的页面也类似于未应用 CSS 的页面，因此，如果将主体内容提前，搜索引擎就能更好地对站点进行索引。最后一点，屏幕阅读器用户也用这种方式访问网站，即按照 HTML 的顺序访问（用户常常在标题之间跳转，而不是听取整个页面，但无论是哪种情况，他们都可以更快地访问到主体内容）。

(3) 以一致的方式使用标题元素（h1 ~ h2），从而明确地标识页面上这些区块的信息，并对它们按优先级排序。

(4) 使用注释来标识页面上不同的区域及其内容。如图 11.2.1 所示，笔者喜欢使用一种不同的注释格式标记区块的开始（而非结束）。

提 示 不一定要在应用 CSS 之前就标记好整个页面。实践中，很少先将一个区块的 HTML 写好，再为其编写一些或全部的 CSS，然后再对下一个区块重复这一过程，等等。处理的方式取决于个人习惯，以及最适合你的方式。对于本章的示例，我先将所有的内容标记好，再对其添加样式。

提 示 你可能已经注意到，我在图 11.2.1 中使用了 section 元素来标记每个包含部分博文的条目。如果它们是完整的博文条目，我就会使用 article 标记它们，就像单独的、完整的博客文章页面那样。那样的话，我可能会聚合完整的博客文章，而不仅仅是包含在示例博文条目中的介绍部分。对它们使用 article 替代 section 也没有错，只是代表这一段代码适合用于聚合。第 3 章包含了各种联合使用 article 和 section 以及单独使用它们的例子。

在 CSS 选择器中使用 ARIA 地标角色（而非 **id**）为元素添加样式

图 11.2.1 为恰当的元素添加了 ARIA 地标角色。在第 9 章中，我曾建议避免或尽可能少地为添加样式而使用 id（参见 9.3 节的“类选择器与 id 选择器的比较”）。

定义 CSS 选择器时，在某些看似需要借助 id 的情况下，其实可以使用地标角色。

以图 11.2.1 中的页脚为例，其 HTML 为：

```
<footer id="footer" role="contentinfo">
   ... page footer content ...
</footer>
```

对应的使用 id 选择器的简化 CSS 为：

```
#footer {
   border-top: 2px dotted #b74e07;
   clear: both;
}

#footer h1 {
   margin-bottom: .25em;
}
```

现在，去掉 id="footer" 使 HTML 得到简化，因为添加这个 id 只是出于为页脚设置样式的目的（而不是为指向此页脚的锚链接所用）。

```
<footer role="contentinfo">
   ... page footer content ...
</footer>
```

地标角色是属性，因此可以在属性选择器中使用它们。

```
footer[role="contentinfo”] {
```

11

```
    border-top: 2px dotted #b74e07;
    clear: both;
}

footer[role="contentinfo"] h1 {
    margin-bottom: .25em;
}
```

第一条样式规则表示“查找 role 属性等于 contentinfo 的 footer 元素”，第二条规则表示“查找**包含在** role 属性等于 contentinfo 的 footer 元素中的 h1 元素”。

结果与前面使用 id 选择器的 CSS 完全相同。

对其他包含地标角色的元素也可以采取类似的做法。对本章示例采取地标角色的做法参见 www.bruceontheloose.com/htmlcss/examples/chapter-11/finished-page-selectors-with-landmark-roles.html。Jeremy Keith 发表的相关讨论参见 http://adactio.com/journal/4267/。

注意，IE6 并不支持属性选择器，因此这种方法可能并不适合你，这取决于你的受众。

重要提示：仅在页面中适合使用地标角色的地方使用它们。不要仅仅为了给某元素设置样式就添加地标角色。这种情况下应该使用类。要复习有关地标角色的知识，参见 3.14 节。

11.3　在旧版浏览器中为 HTML5 元素添加样式

我们已经知道，HTML5 引入了一些新的语义化元素，其中大多数在第 3 章和第 4 章作了讲解。在大多数情况下，现代浏览器原生支持这些元素。从样式的角度来说，这意味着浏览器将为这些新的元素应用默认样式，就像它们对待早在这门语言诞生之际就存在的元素那样。例如，article、aside、nav 和 section（以及其他的一些）元素显示为单独的行，就像 div、blockquote、p 以及其他的在 HTML5 之前的 HTML 版本中称做块级元素的元素。

你可能会想：“对于旧版浏览器呢？这些浏览器出现时还不存在 HTML5 的新元素，我该如何使用这些元素呢？”

不过，值得庆幸的是大多数浏览器允许对它们并不原生支持的元素添加样式。Internet Explorer 是个例外，不过第 (2) 步中描述了一个简单的解决办法。因此，跟着下面三个简单的步骤，就可以开始为包含 HTML5 元素的页面添加样式了。

针对全部浏览器为 HTML5 新元素添加样式

(1) 将下面的代码添加到网站的主样式表文件（即所有页面都用到的样式表文件）：

```
article, aside, figcaption,
➝ figure, footer, header,
➝ hgroup, menu, nav, section {
    display: block;
}
```

原因：大多数浏览器默认将它们无法识别的元素作为行内元素处理。因此这一小段 CSS 将强制 HTML5 新的“类块级”语义元素显示在单独的行（对 IE 还需要做更多的工作，

下一步将会讲到）。各浏览器内置的默认样式表均对 div、blockquote、p 等元素声明了 display: block;。

(2) 对于 Internet Explorer 在 IE9 之前的版本，要为新的 HTML5 元素正常添加样式，将下面突出显示的代码添加到每个页面的 head 元素（不是 header 元素），最好置于指向 CSS 文件的链接后面。

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>photobarcelona</title>
<link rel="stylesheet" href=
→ "assets/css/base.css" />
<!--[if lt IE 9]>
   <script src="http://html5shiv.
   → googlecode.com/svn/trunk/
   → html5.js"></script>
<![endif]-->
</head>
<body>
...
```

要了解为什么要这样做，参见“关于 HTML5 剃刀”。

(3) 现在，可以随意使用 CSS 添加样式了！

在为元素设置样式的过程中，你可能偶尔会遇到一些小故障，不过在大多数时候不会出现问题。

这种方法及替代方法有一个缺陷。由于在 IE6 至 IE8 中为 HTML5 元素设置样式需要 JavaScript，因而如果用户的浏览器不支持或关闭了 JavaScript，这些用户看到的 HTML5 元素就是无样式的，甚至可能是凌乱的。对于你自己的网站，你可能愿意承担这种风险，就像其他很多设计人员和开发人员那样。不过，如果这是你为客户做的网站，在使用需要显示为块级元素的 HTML5 元素之前，需要让客户弄清楚这一点。他们可能有包含浏览器使用情况的用户分析工具以帮助决策。

替代方法

- 如果你对使用 article、section、nav 等第 (1) 步中列出的元素感到不舒服，可以使用 div 替代它们。这也是 HTML5 出现之前人们建立网站的方式。这样做会降低网站的语义化程度，不过这也是一种可接受的方法。有的人甚至选择类来模仿新的 HTML5 元素名称以适应新元素，例如：

  ```
  <div class="header">...</div>
  <div class="article">...</div>
  <div class="section">...</div>
  ```

  等等。
- 如果你确实要使用新的元素，你可以尽可能多地编写指向其他元素的 CSS 选择器，以降低 JavaScript 关闭时产生的影响。我在本章的页面布局中经常这样做。在关闭了 JavaScript 的 IE6 中，页面看起来会有所不同，但也并非不可用，参见图 11.3.1。

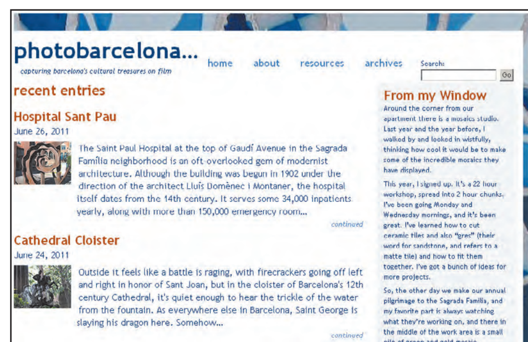


图 11.3.1　在关闭了 JavaScript 的 IE6 中显示的页面。由于关闭了 JavaScript，因而 HTML5 剃刀文件不会被执行。报头区域看起来有点不连贯，但页面从整体上看是完好的

提示　可以下载 html5.js（HTML5 剃刀文件）并添加到网站的文件中去，而不是指向 http://html5shiv.googlecode.com/svn/trunk/html5.js。不过，该文件时常更新，因此，在第 (2) 步中使用有关代码时，从 Google Code 加载这个文件并不是个坏主意。

关于 HTML5 剃刀

与其他主流浏览器不同，Internet Explorer 8 及之前的版本会忽略它们不原生支持的元素的 CSS。

幸好有办法让 IE 的这些版本识别这些元素——使用 JavaScript 为每个元素设置 `document.createElement("elementname")`。这是 Sjoerd Visscher 发现的一种方法。例如，`document.createElement("aside")` 让 IE 可以识别 `aside` 元素，这样，上面第 (1) 步中添加的样式规则（以及你创建的任何其他样式）就生效了。

John Resig 记录了这种方法，并将其命名为 HTML5 剃刀（也称 HTML5 垫片）。谢天谢地，我们不必亲自为每个新的 HTML5 元素编写 JavaScript，因为 Remy Sharp 将 John 的方法打包成了一个 JavaScript 文件，并放在网络社区上（html5shiv.googlecode.com/svn/trunk/html5.js）。后续的贡献者又增强了该文件。

使用 HTML5 剃刀再简单不过了。只需要链接到该文件就可以了，就像第 (2) 步中粗体显示的代码那样。

HTML5 剃刀也被集成进了其他一些 JavaScript 库，如 Modernizr（www.modernizr.com/）。因此，如果你在页面中使用了 Modernizr，就不需要单独加载 HTML5 剃刀了。顺便说一下，Modernizr 是一个非常方便的库，它可以检测浏览器是否支持 HTML5 和 CSS3 的各种特性。下载下来用吧！

11.4　对默认样式进行重置或标准化

前面提到，每个浏览器都有内置的默认样式表。HTML 会遵照该样式表显示，除非你自己编写的 CSS 覆盖了它们。整体上，不同浏览器提供的默认样式表是相似的，但也存在一定的差异。为此，开发人员在应用他们自己的 CSS 之前，常常需要抹平这些差异。

有两种主要的抹平差异的方法。

- 使用 CSS 重置（reset）开始主样式表，如 Eric Meyer 创建的 Meyer 重置（http://meyerweb.com/eric/tools/css/reset/）。另外还有其他的一些重置样式表。
- 使用 Nicolas Gallagher 和 Jonathan Neal 创建的 normalize.css 开始主样式表。该样式表位于 http://necolas.github.com/normalize.css/（点击 Get the normalize.css file 链接）。

CSS 重置可以有效地将所有默认样式都设为“零”，如图 11.4.1 所示。第二种方法，即 normalize.css，则采取了不同的方式。它并不对所有样式进行重置，而是对默认样式进行微调，使它们在不同的浏览器中有相似的外观，参见图 11.4.2。



图 11.4.1　这是应用了重置的示例页面。最大的变化是所有的字体大小都变成一样的了，所有的外边距和内边距都设成 0 了

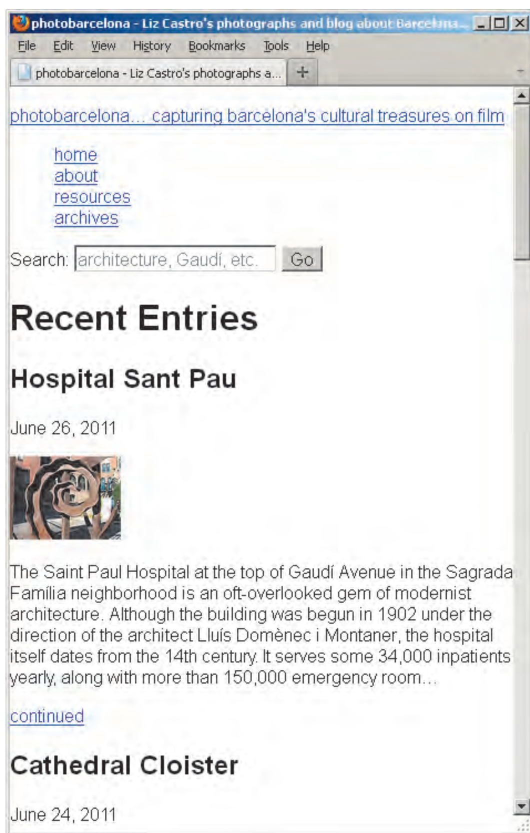


图 11.4.2　这是使用了 normalize.css（而不是重置）的示例页面。它与无样式的默认显示效果很相似，但也有差异。更重要的是，这个版本与使用当今常见的浏览器查看的效果是极为相似的

并非一定要用到这两种方法中的一种。保留浏览器的默认样式，并自己编写相应的 CSS 也是很好的。

在本章中，我使用 Meyer 重置开始编写页面，并对文本添加了一些样式。因此，在应用余下的样式之前，页面如图 11.4.3 所示。由于使用了重置，你可以更清晰地看到本章介绍的 CSS 是如何影响布局的。在阅读本章内容的过程中，你需要了解如何在用到重置的情况对页面进行布局。由于这种方法非常流行，因此这也是一项重要的技能。



图 11.4.3　这是使用了重置并对文本应用了一些样式的示例页面。可以从这里起步，随着本章的介绍逐步为页面添加余下的样式

11.5　盒模型

CSS 处理网页时，它认为每个元素都包含在一个不可见的盒子里。盒子由内容区域、内容区域周围的空间（内边距，padding）、内边距的外边缘（边框，border）和边框外面将元素与相邻元素隔开的不可见区域（外边距，margin）。这类似于挂在墙上的带框架的画，其中衬边是内边距，框架是边框，而该画框与相邻画框之间的距离则是外边距，参见图 11.5.1。

可以使用 CSS 确定每个元素的盒子的外观和位置，并由此控制网页的布局，参见图 11.5.2。

正如第 1 章中讨论的，每个元素的盒子可以是块级的（从新的一行开始，就像一个新的段落），也可以是行内的（不会产生新行）。这对网页初始布局的影响是：在默认情况下，元素按照它们在 HTML 中自上而下出现的次序显示（这被称作文档流，document flow），并在每个块级元素的开头和结尾处换行。

对元素的盒子进行定位有四种基本方法：可以让盒子处于文档流中（这是默认的方式，

也称做静态方法，它也是用得最多的方法，参见图 11.5.3 和图 11.5.4），可以让盒子脱离文档流，并指定该元素相对于其父元素（绝对方法，需慎用）或浏览器窗口（固定方法，实践中用得更少）的精确坐标，还可以相对于盒子在文档流中的默认位置对其进行移动（相对方法，使用频率介于静态方法和另两种方法之间）。此外，如果盒子相互重叠，还可以指定它们的叠放次序（z-index）。

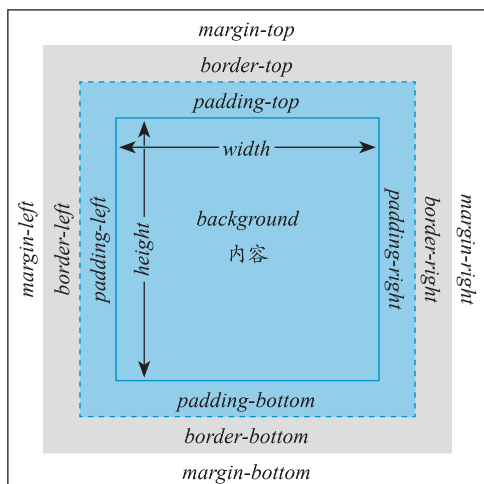


图 11.5.1　每个元素有盒子都有四个决定其大小的属性：内容区域、内边距、边框和外边距。可以单独控制每一个属性

同时，可以控制盒子的外观，包括 `background`、`padding`、`border`、`margin`、`width`、`height`、`alignment`、`color` 等。本章将讨论所有这些属性。

注意，有些布局属性（尤其是 em 值和百分数）是依赖于其父元素的。记住，父元素是直接包含当前元素的元素（参见 1.3 节中的“父元素和子元素”小节）。

提 示　图 11.5.1 是根据 Rich Hauck 的盒模型图绘制的（该图也是根据 CSS 规范中的插图绘制的）：www.mandalatv.net/itp/drivebys/css/。

内容区域与边框之间的空间是内边距（图中四边的内边距均为10像素）。上述区域均会填充背景色。

侧栏内容区域。在这个例子中，没有在CSS中显式地指定宽度和高度。



外边距是边框外边的不可见区域（图中的左外边距为包含整个页面的容器宽度的72%）。

外边缘是边框（这个盒子没有边框）。

图 11.5.2　图 11.5.1 的盒模型

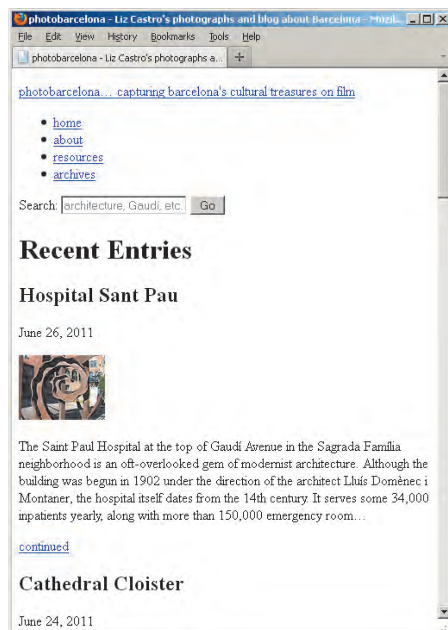


图 11.5.3　还记得本章开头未应用任何样式的示例页面吗？这是示例页面的文档流。尽管为文本添加了样式，但各元素的次序并未改变，参见图 11.5.4



图 11.5.4 这是目前为止页面的显示效果。由于仅对文本添加了样式，因此常规的文档流没有任何变化

11.6 修改背景

正如 10.9 节讲到的，可以对整个页面和单个元素设置背景，如图 11.6.1 所示。这包括几乎所有的元素，甚至是表单和图像（是的，图像也可以有背景图像！）。

1. 使用背景图像

(1) 输入 background-image:。

(2) 输入 url(*image.png*)，这里的 *image.png* 是要用做背景的图像的路径和文件名（如图 11.6.1 所示）。或者输入 none，表示不使用任何图像（只能用于覆盖另一条为元素应用背景图像的样式规则）。

2. 让背景图像重复显示

输入 background-repeat: *direction*，这里的 *direction* 可以是 repeat（使图像在水平、垂直两个方向平铺）、repeat-x（使图像在水平方向平铺）、repeat-y（使图像在垂直方向平铺），参见图 11.6.1；或 no-repeat（不平铺图像）。

3. 控制图像是否随页面一起滚动

(1) 输入 background-attachment:。

(2) 输入 fixed，表示让背景图像固定在浏览器窗口中（即访问者滚动页面时背景图像保持不动），或者输入 scroll，表示让背景图像随着页面的滚动而移动。scroll 是默认值，因此，如果这是你需要的效果，就不必指定。这是很常见的情形。

4. 指定背景图像的位置

输入 background-position: *x y*，其中 *x*、*y* 可以是百分数，也可以是绝对值，如 20px 147px（允许输入负数）。或者对 *x* 使用 left、center 或 right，对 *y* 使用 top、center 或 bottom。（更多例子见 10.9 节中的“更多关于背景的说明”。）

5. 修改背景颜色

(1) 输入 background-color:。

(2) 输入 transparent（让元素的背景露出来）或 *color*，这里的 *color* 是颜色名称、十六进制数颜色、RGB 颜色、RGBA 颜色、HSL 颜色或 HSLA 颜色（参见 7.4 节中的“CSS 颜色”），如图 11.6.4、图 11.6.5 和图 11.6.6 所示。

6. 同时修改所有背景属性

(1) 输入 background:。

(2) 以任意顺序指定任何可接受的背景属性值（如“使用背景图像”到“修改背景颜色”所述）参见图 11.6.2 和图 11.6.4。

提 示 background-color 的默认值是 transparent，background-image 的默认值是 none，background-repeat 的默认值是 repeat，background-attachment 的默认值是 scroll，background-position 的默认值是 top left（同 0 0）。

```
#container {
background-image:
→ url(../img/bg-bluebench.jpg);
    background-repeat: repeat-y;
}
```

图 11.6.1　首先，对最外层的容器（包含 id="container" 的 div）应用背景图像。再在垂直方向（y 轴）重复图像。这是定义背景图像的长格式——每个属性都单独声明。为了让代码尽可能地紧凑，我在样式表中使用了简记法，如图 11.6.2 所示。但不管使用哪种方法，图 11.6.3 中的实现结果都不太好

```
[code block figure]
#container {
background: url(../img/bg-bluebench.jpg)
→ repeat-y;
}
```

图 11.6.2　可以按照第 10 章和稍后的介绍，使用 background 简记法一次性应用多个与背景有关的属性。推荐在任何时候都尽可能使用简记法，尽管有时候指定单个属性也是有意义的。此外，背景图像的路径写做 ../img/bg-bluebench.jpg，这是因为样式表位于与 /img/ 文件夹并列的文件夹。指向你自己的图像的路径可能并不相同



图 11.6.3　这个页面现在看起来有点恐怖，背景图像让文本变得很难辨认。最终，我们会覆盖这个背景图像，让文本重新变得可读。之后再将掀开这层覆盖层，露出背景的某些部分

```
#container {
background: url(../img/bg-bluebench.jpg)
repeat-y;
}

#page {
background: #fef6f8;
}

a:focus,
a:hover,
a:active {
background: #f3cfb6;
}

.logo a:hover {
    background: transparent;
}

.sidebar {
background: #f5f8fa;
}
```

图 11.6.4　这些地方使用 background-color 属性也可以，但没有必要，因此我再次使用了 background 简记法。为 page div 设置的背景会让文本变得可读（在本章稍后部分，我会将这个颜色改成白色）。接着，为链接在鼠标停留时的状态添加了背景，从而更清楚地表明它们是链接。同时，对带链接的网站标识覆盖了这个效果，为它赋予了透明的背景，从而在鼠标停留时背景图像也能露出来。最后，为 sidebar div 中的每月评论栏添加了背景颜色。这里使用了 class，因此，如果将来另一个侧栏需要类似的样式就很容易了

提 示　使用 background 简记法时，不一定要指定所有的属性。但是要注意，任何没有指定的属性都会设为默认值，它们可能覆盖先前定义的样式规则。

提 示　background 属性不是继承的。如果想覆盖另一个样式规则，只需要显式地设置像 transparent 或 scroll 这样的默认值。



图 11.6.5　page div 的背景完全覆盖了图 11.6.3 中设置的背景图像。稍后将使用内边距改变这种情况。注意，没有鼠标停留的链接也具有与 page 相同的背景，而鼠标停留的 About 链接则具有高对比度的背景，以提醒访问者注意



图 11.6.6　sidebar 的背景颜色是最浅的蓝色，这足以使它与其他部分有所区别（另见彩插）

提示　如果使用 background-position 并设为 repeat，那么这个位置指定的是重复的图像中第一个图像开始的位置。例如，从 top right、0 20px 等位置开始重复。

提示　可以在 background-position 属性中使用负数。例如，background-position: -45px 80px 会将图像定位在元素左边外侧 45 像素（因此不会看到图像在水平方向上的头 45 个像素）、顶部内侧 80 像素。

提示　要为整个页面设置背景，可以为 body 元素设置 background 属性。

提示　如果同时指定背景颜色和 URL，在 URL 中的图像加载时才能显示这个颜色，并在背景图像的透明部分及图像未覆盖到的部分露出这个颜色。

提示　要仔细选择文本和背景的颜色（及背景图像），使两者具有足够高的对比度。这对视障用户来说尤其重要。

11.7　设置元素的高度和宽度

可以为很多元素设置高度和宽度，如分块内容、段落、列表项、div、图像、video、表单元素等（参见图 11.7.1 和图 11.7.2）。同时，可以为短语内容元素（默认以行内方式显示）设置 display: block; 或 display: inline-block;，再对它们设置宽度或高度。（关于 display 属性的更多信息，参见 11.19 节。）

1. 设置元素高度或宽度的步骤

(1) 输入 width: *w*，其中的 *w* 是元素内

容区域的宽度，可以表示为长度（带单位，如 px、em 等）或父元素的百分数。或者使用 auto 让浏览器计算宽度（这是默认值）。

(2) 输入 height: *h*，其中的 *h* 是元素内容区域的高度，只能表示为长度（带单位，如 px、em 等）。或者使用 auto 让浏览器计算高度（这是默认值）。

```
#container {
    background: url(../img/bg-bluebench.jpg)
    → repeat-y;
    width: 90%;
}

#page {
    background: #fef6f8;
    width: 97.9167%;
}

#main {
width: 71%;
}

input[type="text"] {
    width: 150px;
}

.photo {
height: 75px;
width: 100px;
}
```

图 11.7.1　将 container div 的宽度限制为浏览器窗口的 90%，可以留出一些空间（即图 11.7.2 中边上的留白），从而不至于显得太挤。通过将 page div 的宽度减少到 container div 的 97.9167%，可以在边上看到一部分背景图像（要了解我如何得出这个百分数，参见本书网站示例代码中的注释；同时，你也可以自由使用其他常规的百分数）。通过将 main div 的宽度设为 page div 的 71%，可以为侧栏留出一些空间（侧栏将在随后移到边上）。input 的样式设置了搜索表单的宽度。最后，.photo 的尺寸控制了博客文章中包在图像周围的段落的大小（.photo 的尺寸等于图像本身的宽度和高度）

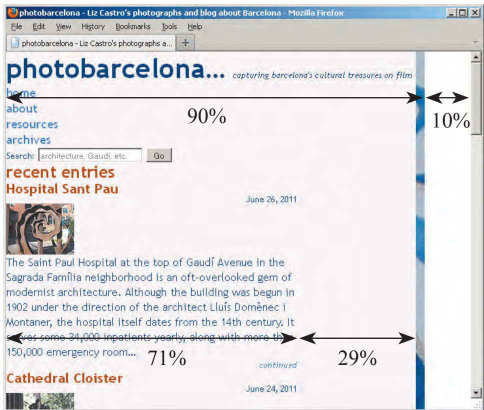


图 11.7.2　container div 包围 page div，它现在的宽度占浏览器窗口的 90%。它的背景图像在边上显示了一部分，这是因为 page div 的宽度也减少了。main div 的宽度是 page div 的 71%，而不是 container div 或浏览器窗口的 71%

```
#container {
background: url(../img/bg-bluebench.jpg)
repeat-y;
max-width: 950px;
width: 90%;
}
```

图 11.7.3　max-width 属性非常适合为流式布局设置外围限制。在我们的例子中，即便访问者的屏幕很大，也不希望内容容器太宽。如果不希望元素太窄，可以应用 min-width 属性，尽管在移动电话和其他较小的设备上浏览万维网已变得越来越普遍，但设置 min-width 仍是一个明智的选择

提 示　如果不显式地设置宽度或高度，就使用 auto（参见下一节）。

提 示　记住，百分数是相对于父元素的宽度的，而不是相对于元素本身的原始宽度的。

提 示　width 的值不包含内边距、边框和外边距（参见下一节）。

提示　不能为显示为行内元素的元素（如短语内容）设置高度或宽度，除非为它们设置 display: inline-block 或 display: block。关于 display 属性的更多信息，参见 11.19 节。

提示　width 和 height 不是继承的。

提示　还有 min-width、min-height、max-width 和 max-height 属性，参见图 11.7.3（如果你的网站使用 Internet Explorer 6，要注意 IE6 并不支持这些属性）。

2. 宽度、外边距和 auto

对于大多数默认显示为块级元素的元素，width 的 auto 值是由包含块的宽度减去元素的内边距、边框和外边距计算出来的。包含块的宽度指的是父元素给元素留出的宽度。

对于图像这样的元素，auto 宽度等于它们固有的宽度，即外部文件的实际尺寸（如本章示例页面中的图像，就是 100 × 75）。浮动元素的 auto 宽度为 0，非浮动的行内元素会完全忽略 width 属性（即不能为 em、cite 等元素设置宽度，除非将它们设置为 display: inline-block 或 display: block）。关于浮动，参见 11.10 节；关于 display，参见 11.19 节。

如果手动设置 width、margin-left 和 margin-right 值，但这些值加上边框和内边距的值不等于包含块的大小，有的属性就需要作出让步。实际上，浏览器会覆盖你对 margin-right 的设置，将其设为 auto（参见图 11.7.4 和图 11.7.5）。

```
div {
background: yellow;
width: 300px;
}

p,
.example {
background: white;
border: 6px solid blue;
margin: 10px;
padding: 5px;
}

.example { /* 第二个段落 */
background: white;
border-color: purple;
width: 200px;
}
```

图 11.7.4　在这个例子中，将父元素 div 的 width 设为 300 像素。这将是我们的包含块。然后，两个段落各个边上都有 10 像素的外边距、5 像素的内边距及 6 像素宽的边框。第一个段落的宽度是自动设置的，因为 auto 是 width 的默认值（除非指定其他的值）。第二个段落（在 HTML 中带有 class="example"）设为 200px

如果手动设置 width，但将某个外边距设为 auto，那么这个外边距将进行伸缩以弥补不足的部分。

不过，如果手动设置 width，并将*左右*外边距都设为 auto，那么两个外边距就将设为相等的最大值（导致元素居中；例如，#container { margin: 20px auto; } 会使对应的元素在页面居中显示）。这也是我对示例页面作的设置，如 11.8 节所示。

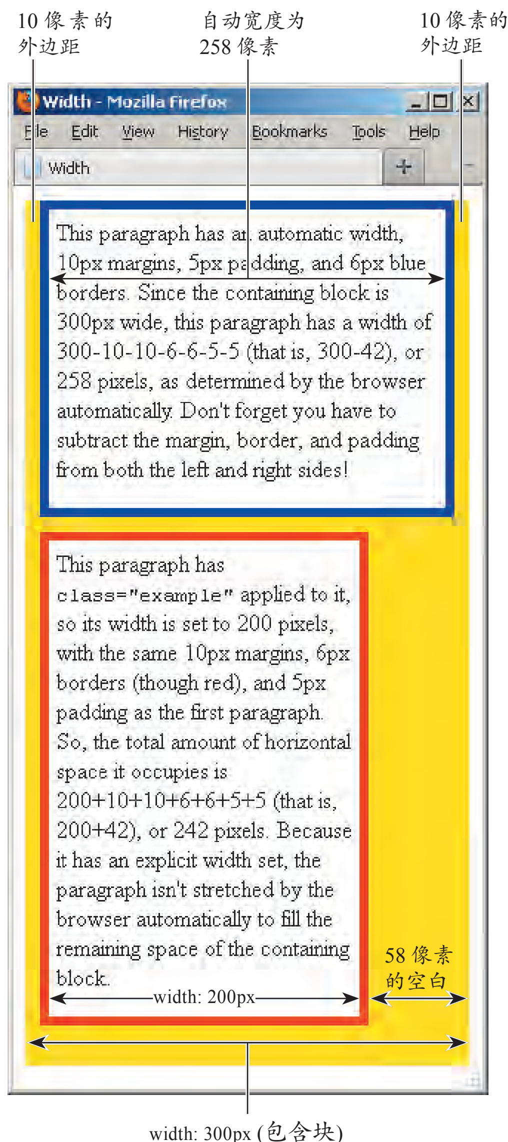


图 11.7.5　如果 width 是 auto（就像第一个段落那样），它的值就是由包含块（黄色区域）的宽度减去元素自身的外边距、内边距和边框计算出来的。如果 width 是手动设置的（就像第二个段落那样），则右外边距常常会进行调整以填补不足的空间

为什么 min-height 通常比 height 更适用

除非你确定元素的内容不会变得更高，最好避免在样式表中指定高度。在大多数情况下，可以让内容和浏览器自动控制高度。这可以让内容在浏览器和设备中根据需要进行流动。

如果设置了高度，随着内容变多，它们有可能撑破元素的盒子，这可能是你预期之外的。在这种情况下，符合标准的浏览器**不会**自动扩大高度，它们在你指定高度时采用了这个指令，并一直坚持下去。（IE6 并不遵循标准，它会扩大高度。）

不过，如果希望元素**至少**具有某一特定高度，可以设置 min-height。如果内容日后变多，元素的高度会自动按需增长。这是 height 与 min-height 的区别，width 和 min-width 也是类似的。

或许你还有所疑虑，实际上，很多原因都会导致内容增长。你的内容或许来自数据库、新闻源或由用户生成的内容。同时，访问者可能增大其浏览器的字体大小，覆盖你指定的样式。

11.8　设置元素周围的外边距

外边距是元素与相邻元素之间的透明空间（参见图 11.8.1 和图 11.8.2）。关于它与元素边框和内边距的关系，参见 11.5 节。

```
#container {
    background: url(../img/bg-bluebench.jpg)
    → repeat-y;
    margin: 20px auto;
    max-width: 950px;
    width: 90%;
}
```

图 11.8.1　主要的外边距调整是对 container div 的。如果对外边距设置两个值，第一个应用于上下外边距，第二个应用于左右外边距。我们将上下外边距设为 20px，为设计留出一些空间。如果为 container 显式地设置了宽度，再将左右外边距设为 auto，就会让页面在浏览器里水平居中显示，参见图 11.8.2



图 11.8.2　将外边距设为 auto 会将 container div 没有使用的占浏览器窗口宽度 10% 的空间平分给左右外边距，从而使布局在窗口里居中。（不必担心落在页面左侧的列表项目符号，我们在本章随后部分将对它们进行处理。）

设置元素外边距的方法

输入 margin: *x*，其中的 *x* 是要添加的空间量，可以表示为长度、相对父元素宽度的百分数或 auto。

提 示　如果对 margin 使用一个值，这个值就会应用于全部四个边。如果使用两个值，那么前一个值会应用于上下两边，后一个值会应用于左右两边（如图 11.8.1 所示）。如果使用三个值，那么第一个值会应用于上边，第二个值会应用于左右两边，第三个值会应用于下边。如果使用四个值，那么它们会按照时钟顺序，依次应用于上、右、下、左四个边（参见图 11.8.3）。

提 示　还可以为 margin 属性添加以下后缀，从而仅将外边距应用于一个边：-top、-bottom、-left 或 -right（如图 11.8.3 所示）。在 margin 和后缀之间不应有任何空格（例如，margin-top: 10px）。

提 示　margin 属性的 auto 值依赖于 width 属性的值（参见 11.7 节）。

```
h1 {
     font-size: 1.5em; /* 24px/16px */
     margin-bottom: .75em;
     text-transform: lowercase;
}

aside h2 {
     font-size: .9375em; /* 15px/16px */
     margin-bottom: 3px;
     text-transform: lowercase;
}

#masthead {
     margin-bottom: 30px;
}

#footer {
     margin-top: 10px;
}

.entry { /* 博客片段 */
     margin: 0 .5em 2em 0;
}

.continued {
     font-style: italic;
     margin-top: -5px;
}

#related { /* 侧栏 */
     margin-left: 72%;
}
```

图 11.8.3　侧栏 div（标记 id="related"）的左外边距为 72%，这比宽度为 71% 的主要内容 div 距离左侧的距离更大。在为侧栏设置浮动之前，它仍会显示在主要内容 div 的下面（如图 11.8.4 所示）。我为其他一些元素也设置了外边距，让它们有一些呼吸空间，如报头下边、页脚上边以及部分博客项目的右边和下边（参见图 11.8.5）

提 示　如果元素位于另一个元素的上面，对于相互接触的两个 margin（即元素相互接触的下外边距和上外边距），仅使用其中较大的一个，另一个外边距会被叠加。左右外边距不叠加。

提 示　外边距不是继承的。



图 11.8.4　侧栏现在距离左边缘的距离为 72%



图 11.8.5　现在，一些元素的周围多出了一些空间

提 示　我为更多元素（比图 11.8.1 和图 11.8.3 更多）添加了外边距，完整的代码见 www.bruceontheloose.com/htmlcss/examples/chapter-11/finished-page.html。

11.9　在元素周围添加内边距

顾名思义，内边距就是元素内容周围、边框以内的空间。根据上文的类比，内边距就像是图画（内容）与画框（边框）之间的衬边。可以改变内边距的厚度（如图 11.9.1、图 11.9.3 和图 11.9.5 所示），不能改变它的颜色或纹理，但元素的背景颜色和背景图像将显示在内边距的空间里（如图 11.9.2、图 11.9.4、图 11.9.6 和图 11.9.7 所示）。

```
#container {
      background: url(../img/bg-bluebench.jpg)
      → repeat-y;
      margin: 20px auto;
      max-width: 950px;
      width: 90%;
      padding: 30px 10px 0 0;
}
```

图 11.9.1　同外边距类似，如果为 padding 设置四个值，那么它们分别表示上、右、下、左（按时钟顺序）内边距。因此在这里只有上边和右边有内边距（实现结果如图 11.9.2 所示）



图 11.9.2　为 container div 添加内边距之后，其外边距和内容（在这个例子中为 page div）之间就产生了一些空间（因为边框默认为 0）。因此，container 背景图像露出来的部分又多了一些

```
#page {
      background: #fef6f8;
      max-width: 940px;
      padding: 10px 10px 10px 0;
      width: 97.9167%; /* 940px/960px */
}
```

图 11.9.3　现在，为 page div 的内容添加内边距。只有上、右、下三个边设置了内边距，左边没有设置

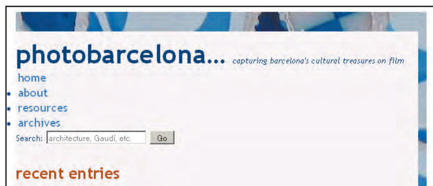


图 11.9.4　我们为 page div 添加了内边距，因此可以注意到 photobarcelona 和上边缘之间多出了 10 像素

```
#page {
     background: #fff;
     padding: 10px 10px 10px 0;
     max-width: 940px;
     width: 97.9167%; /* 940px/960px */
}

.sidebar {
     background: #f5f8fa;
     padding: 10px;
}

.archive ol {
     /* 将列表项目由数字改为圆点 *
        list-style: disc;
     /* 对圆点进行缩进  */
     padding-left: 18px;
}
```

图 11.9.5　我将页面的临时背景颜色改成了白色（#fff）（如图 11.9.6 所示）。同时，我为侧栏的四个边都添加了一些内边距，让内容不至于挨到边上；为有序列表的左边添加了内边距，让列表项目符号有一些缩进（如图 11.9.7 所示）。（关于列表的更多信息，参见第 15 章。）

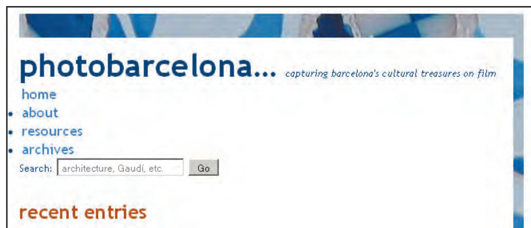


图 11.9.6　当 page 的背景设成白色时，就可以很清楚地看到为什么左边不需要添加内边距。我还做了其他一些内边距调整，读者可以在本书网站的代码文件中找到

在元素周围添加内边距的方法

输入 padding：*x*，这里的 *x* 是要添加的空间量，表示为带单位（通常为 em 或像素）的长度或父元素宽度的百分比（如 20%）。

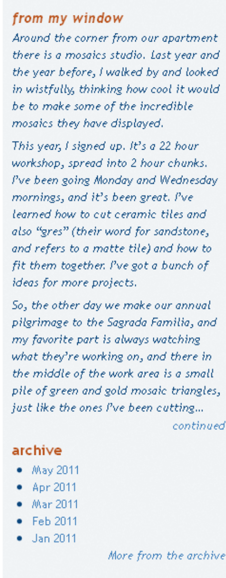


图 11.9.7　侧栏的背景颜色显示在图 11.9.5 中四个边上添加的 10 像素宽的内边距空间里

提 示　同 margin 属性一样，对于 padding 属性，如果使用一个值，那么指定的内边距将完全一样地应用于四个边（如图 11.9.5 所示）。如果使用两个值，那么第一个值将应用于上下内边距，第二个值将应用于左右内边距。如果使用三个值，那么第一个值将应用于上内边距，第二个值将应用于左右内边距，第三个值将应用于下内边距。如果使用四个值，那么它们将按照时钟顺序依次应用于上、右、下、左内边距（如图 11.9.1 和图 11.9.3 所示）。

提 示　还可以为 padding 属性添加以下后缀，从而仅将内边距应用于一个边：-top、-bottom、-left 或 -right。在 padding 和后缀之间不应有任何空格（例如，padding-right: 1em）。

提 示　内边距不是继承的。

11.10　使元素浮动

可以使元素浮动在文本或其他元素上。可以使用这种技术让文本环绕在图像周围（如图 11.10.1 和图 11.10.2 所示），或者创建多栏布局，等等。

```
.photo {
    float: left;
    height: 75px;
    width: 100px;
}
```

图 11.10.1　如果让元素浮动到一边，内容通常会显示在文档流中该元素后面的位置。这里，我让图像容器（带 class="photo" 的 p 元素）向左边浮动，从而让博客条目的介绍文本围绕在它旁边。（也可以直接向 img 元素应用 float 属性。）如图 11.10.2 所示，如果文本的高度大于图像的高度，文本会环绕在图像周围。这种效果在很多情况下是符合预期的，但对于这个页面，我想让文本垂直显示下去，不管它有多长。为此，我为文本添加了一定的左外边距（参见图 11.10.3）



图 11.10.2　由于 .photo 容器向左浮动，文本会沿着它的周围进行排列。随着文本不断变长，它会环绕在图像的周围

```
/* 这个类位于既包含介绍性文字，又包含
→ continue链接的div上 */
.intro {
    margin: -5px 0 0 110px;
}
```

图 11.10.3　通过为包围文本的容器设置 110px 的左外边距，文本将显示在远离左边缘的位置，哪怕它的高度大于图像的高度。因此，文本将不再环绕在图像的周围（参见图 11.10.4）。此外，我还为文本添加了 -5px 的上外边距，从而让它与左侧图像的上边缘对齐



图 11.10.4　介绍文字不再进行环绕。接下来会使 main div 浮动（参见图 11.10.5），从而让侧栏显示在它的旁边（如图 11.10.6 所示）

让文本环绕元素

(1) 输入 float:。

(2) 输入 left，让元素浮动到左边，其他内容围绕在它右边（参见图 11.10.1 ~ 图 11.10.7）；

或者输入 right，让元素浮动到右边，其他内容围绕在它左边；

或者输入 none，让元素完全不浮动。（none 是默认值，因此只有在特定情况下需要覆盖某个浮动元素的样式规则时才需要用到它。）

(3) 使用 width 属性显式地设置元素的宽度（参见 11.7 节）。

```
#main {
    float: left;
    width: 71%;
}

/* 我们之前已经对侧栏设置了这个外边距 */
#related {
    margin-left: 72%;
}
```

图 11.10.5 现在，使用相同的方法让主体内容向左浮动，从而让侧栏显示在它旁边。（前面通过为侧栏设置左外边距，已将它推到右侧，但仍显示在主体内容的下方。）



图 11.10.6 这里，main div 向左浮动，因此侧栏得以靠在 main div 的右侧。事实上，页脚也会这样（参见图 11.10.7），因为它在 HTML 中紧跟在侧栏后面。（在下一节里，我们会让页脚回到它该有的位置。）注意，并不一定要给侧栏（#related）设置很大的左外边距以产生浮动效果，正如让博客文本浮动在图像容器周围时也不必这样做（参见图 11.10.1 和图 11.10.2）。但是类似于图 11.10..3 和图 11.10.4 中避免环绕的做法，设置左外边距可以防止在侧栏文本的高度大于 main div 的高度时侧栏内容环绕在 main div 的下方。同时，如果不为侧栏设置左外边距，它的背景颜色将平铺到整个 main div

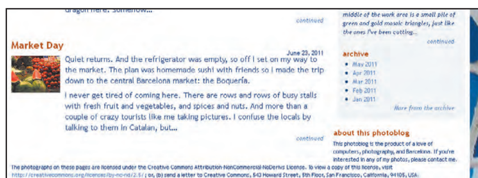


图 11.10.7 由于 main div 向左浮动，因此所有其他的元素（包括页脚）都会环绕在它的周围，除非进行其他设置。下一节将对这些设置进行讲解

提 示 记住，你选择的方向是应用于需要浮动的元素的，而不是应用于环绕它的元素的。在使用 float:left 时，页面的其他部分围绕在右边，反之亦然。

提 示 对于某些元素（如短语内容），如果不显式地设置宽度，它们将无法正确地浮动。

提 示 float 属性不是继承的。

11.11 控制元素浮动的位置

可以控制元素能够浮动在哪些元素的旁边，以及不能浮动在哪些元素的旁边。要阻止元素浮动在不合适的元素旁边，可以使用 clear 属性。

控制元素浮动位置的步骤

(1) 输入 clear:（如图 11.11.1 和图 11.11.2 所示）。

(2) 输入 left，阻止元素浮动在该元素的左边（参见图 11.11.3）；

输入 right，阻止元素浮动在该元素的右边；

输入 both，阻止元素浮动在该元素的左右两边；

输入 none，允许元素浮动在该元素的左右两边。

```
#footer {
    clear: both;
    margin-top: 10px;
}
```

图 11.11.1　在 11.10 节可以看到页脚环绕在浮动的 main div 周围。这里，为页脚设置 clear: both; 以阻止上述情况（参见图 11.11.2）。也可以使用 clear: left; 因为唯一需要担心的浮动元素是向左浮动的。但是，对两边都进行清理并没有坏处，而且，随着设计越来越复杂，这样做也比较方便

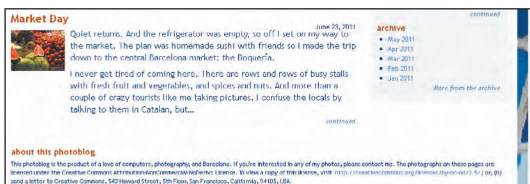


图 11.11.2　clear 属性指示的是设置该属性的元素（在这个例子中是页脚）不能环绕在浮动元素的旁边，而要显示在浮动元素的后面

```
.logo {
    float: left;
    font-size: 2.5em; /* 40px/16px */
    margin: 0;
}

/* 这个div既包围主导航，又包围搜索表单 */
#masthead div {
    float: right;
}

/* :::: 站点导航 :::: */
.nav li {
    float: left;
    font-size: .75em; /* 12px/16px */
    padding: 0 25px 0 3px;
}
```

图 11.11.3　在为报头设置样式时，标志（包括标语）是向左浮动的。包围主导航和搜索框的 div 是向右浮动的。在该 div 中，每个导航列表项目都是向左浮动的，从而让每个项目都可以显示在相邻项目的旁边，而不是竖直排列

提　示　应该将 clear 属性添加到不希望环绕浮动对象的元素上（参见图 11.11.4、图 11.11.5 和图 11.11.6）。因此，如果要让一个元素在右侧没有浮动元素（以及任何靠向右侧的元素）之后才显示，就为它添加 clear: right;（而不是为浮动的元素添加此样式规则）。

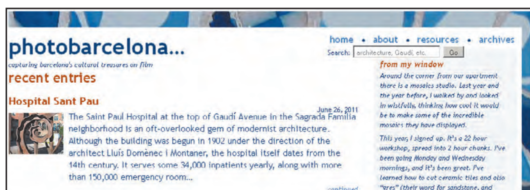


图 11.11.4　大部分报头的布局都是好的，只是报头下面的内容（特别是侧栏）因为 float 而向上移动了一些。这是因为报头包含的元素（header）的高度变小了一些，因为计算元素高度时不把其中的浮动元素包含在内

```
<div id="container">
    <div id="page">
        <header id="masthead" role="banner"
        → class="clearfix">
            ...
        </header>

        <div id="main" role="main">
            ...
        </div>
```

图 11.11.5　为清除报头的 float，从而让主体内容和侧栏 div 不紧靠着它，可以在主体内容和侧栏外面再添加一层 div，并为其设置 clear: both; 不过，要让 HTML 尽可能简洁，这样做并不理想。另一种流行的解决方案是使用 clearfix。使用这种方法，需要做的只是为报头应用 class="clearfix"。假定在样式表中已经为 .clearfix 设好了样式，问题就解决了（参见图 11.11.6）。更多细节信息参见“清除 float 的其他方法”



图 11.11.6 clearfix 方法很神奇地消除了报头的 float，从而让跟在它后面的内容显示在它的下方

提示 页面中的标语由一个 span 包围。样式表中有一条规则是 .logo span { display: block; }（但没有将其显示出来）。这条规则让 span 显示在单独的行，就像段落等默认显示为块级的元素。更多信息参见 11.19 节。

清除 float 的其他方法

要清除报头的 float，从而让主体内容和侧栏内容不靠向它（参见图 11.11.4），还有其他一些方法。

overflow 方法

第一种也是最简单的方法就是将下面的代码添加到样式表中：

```
#masthead {
  overflow: hidden;
}
```

（overflow 属性将在本章后面讲到。）有时使用 overflow: auto; 也有效，但这样做可能会出现一个滚动条，这显然是不希望看到的。在某些情况下，overflow: hidden; 会将内容截断，对此要多加注意。对于我自己的网站，我仅在 overflow 能解决 float 问题的情况下才使用 overflow 方法（就像这个报头的问题就是这种方法能解决的）。如果这种方法不能解决，就使用 clearfix（参见图 11.11.5）。通常，使用 clearfix 方法更容易保持一致，这也是我为什么在示例中展示这种方法的原因（尽管它是我最后考虑的方法）。

clearfix 方法

多年以来，由于 Web 社区的贡献，.clearfix 的 CSS 已经经历了好几个不同的版本。下面的版本来自出色的 HTML5 Boilerplate（www.html5boilerplate.com）。该项目由 Paul Irish 发起，随后大量社区开发人员为其贡献了代码。推荐你下载使用。

该项目核心成员 Nicolas Gallagher 贡献了下面的 clearfix 代码。将它们复制到你的样式表中，再为包含 float 的元素添加 **class="clearfix"** 即可（参见图 11.11.5）。

```
.clearfix:before, .clearfix:after { content: ""; display: table; }
.clearfix:after { clear: both; }
.clearfix { zoom: 1; }
```

这段代码有一点复杂，因此这里不再多作解释。相关的解释与讨论参见 http://nicolasgallagher.com/micro-clearfix-hack/（注意，这些讨论都很技术）。

小结

使用 clear 属性和使用 overflow 或 clearfix 方法有一些细微的差别。使用 clear 时，该属性应用于不想环绕浮动元素的元素。对于另两种方法，有关样式则应用于浮动元素的容器或元素本身。

11.12　设置边框

可以在元素周围创建一个边框，并设置它的厚度、风格和颜色，参见图 11.12.1。如果指定了内边距（参见 11.9 节），边框将包围在元素内容和内边距的外面。

1. 定义边框风格

输入 border-style: type，其中的 type 可以是 none、dotted（点状）、dashed（虚线）、solid（实线）、double（双线）、groove（凹槽）、ridge（凸槽）、inset（凹边）或 outset（凸边），参见图 11.12.1。

```
#masthead {
     border-bottom: 2px dotted #1d3d76;
     margin-bottom: 30px;
     padding-bottom: 20px;
}

.entry {
     border-right: 2px dashed #b74e07;
     margin: 0 .5em 2em 0;
}

#footer {
     border-top: 2px dotted #b74e07;
     clear: both;
     margin-top: 10px;
}
```

图 11.12.1　通过设置边框可以为设计增添一些亮点。在这个例子中，在报头下边添加点状边框（参见图 11.12.2）就可以起到这个作用，同时还能帮助访问者将报头同页面其余部分快速区分开来

2. 设置边框宽度

输入 border-width: *n*，这里的 *n* 是需要的宽度（含单位，如 4px）。

3. 设置边框颜色

输入 border-color: *color*，这里的 *color* 是颜色名称、十六进制值或 RGB、HSL、RGBA、HSLA 颜色（参见 7.4 节的“CSS 颜色”）。



图 11.12.2　注意，由于虚线右边框应用于每个 entry section 元素而不是 main div 元素，因此每个条目之间都有中断。不同于报头和页脚的边框（参见图 11.12.3），这里用的是虚线

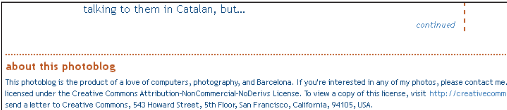


图 11.12.3　页脚上边框采用了同报头下边框相同的样式（点状），只是颜色不一样。同报头的下边框一样，它也能起到从视觉上区分页脚与其他内容的作用

4. 使用简记法同时设置多个边框属性

(1) 输入 border。

(2) 如果需要，输入 -top、-right、-bottom 或 -left 将边框效果限制在某一条边上。

(3) 如果需要，输入 -*property*，这里的 *property* 可以是 style（风格）、width（宽度）或 color（颜色），仅使用单个属性设置边框。

(4) 输入冒号（:）。

(5) 输入恰当的值（如前面三小节介绍的那样）。如果跳过了第 (3) 步，则可以指定所有三种边框属性，也可以指定三种类型的任意组合（如 border:1px solid 和 border-right: 2px dashed green;）。如果在第 (3) 步中指定了一种属性，则只能使用这种属性可以接受的值（如 border-right-style: dotted;）。

提 示 边框不是继承的。

提 示 各个边框属性(border-width、border-style 和 border-color)均可接受一至四个值。如果使用一个值，那么它会应用于全部四个边。如果使用两个值，那么前一个值会应用于上下两边，后一个值会应用于左右两边。如果使用三个值，那么第一个值会应用于上边，第二个值会应用于左右两边，第三个值会应用于下边。如果使用四个值，那么它们会按照时钟顺序，依次应用于上、右、下、左四个边。

```
p {
    border: 10px solid red;
    padding: 15px;
}

p.ddd {
    border-width: 4px;
    border-style: dotted dashed double
    → solid;
}

p.inset {
    border: 10px inset blue;
}

p.outset {
    border: 10px outset green;
}

p.groove {
    border: 10px groove purple;
}

p.ridge {
    border: 10px ridge orange;
}
```

图 11.12.4 在这个例子中，我为每个段落设置了内边距和默认边框。然后，对第一个段落设置了应用于四个边的边框宽度，并为每个边设置了不同的风格。对于余下的四个段落，使用一条声明并在其中重复 10px 比通过两个属性分开设置风格和颜色要方便一些

提 示 要让边框显示出来，至少必须定义边框风格（参见图 11.12.4 和图 11.12.5）。如果没有定义风格，就不会显示边框。风格的默认值是 none。



图 11.12.5 各浏览器对边框风格的处理方式并不完全相同。这是在 Firefox 中显示的不同边框风格，从中可以了解到不同风格的差异（另见彩插）

提 示 如果使用简记法，如 border 或 border-left 等，则未提供的属性将显示其默认值。因此，border: 1px black; 相当于 border: 1px black none;，这意味着不会显示边框（即使在之前使用 border-style 指定了边框风格）。

提 示 边框的默认颜色是元素 color 属性的值（参见 10.8 节）。

提示　IE7 及以下版本无法显示 groove、ridge、outset、inset 等双色调边框风格。它们会显示为 solid 风格。

提示　border 属性可用于表格及其单元格。

提示　CSS3 引入了 border-image 属性。除 Internet Explorer 以外，浏览器对它的支持程度较好（参见 http://caniuse.com/#search=border-image）。关于 border-image，参见 www.sitepoint.com/css3-border-image/。

11.13　偏移自然流中的元素

每个元素在页面的文档流中都有一个自然位置（参见图 11.13.1）。相对于这个原始位置对元素进行移动就称为相对定位（如图 11.13.2 和图 11.13.3 所示）。周围的元素完全不受此影响（如图 11.13.3 所示）。

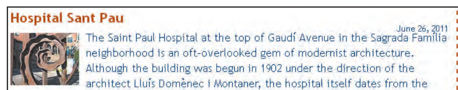


图 11.13.1　尽管日期是右对齐的，但它与标题位于不同的行，出现在标题的下方，而且它与博客文本的距离也太近了

```
.entry .date {
    line-height: 1;
    margin: 0 1em 0 0;
    padding: 0;
    position: relative;
    top: -1em;
}
```

图 11.13.2　记住，既要指明采用相对位置，还要给出偏移量。可以使用正数，也可以使用负数。使用 em 会使偏移量的大小与文本字体大小成比例。由于 1em 等于元素的字体大小，因此在这个例子中，样式声明会让日期向上移动一小段距离（因为设置了 -1em），参见图 11.13.3

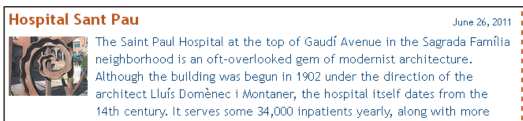


图 11.13.3　通过对日期设置负数偏移量，使它上升到前一个块的空间里。在这个例子中，这样做使日期与标题对齐。后续的元素完全不受影响

偏移自然流中元素的步骤

(1) 输入 position: relative;（不要忘了输入分号；，空格是可选的）。

(2) 输入 top、right、bottom 或 left。再输入 :*v*，这里的 *v* 是希望元素从它的自然位置偏移的距离，可以表示为绝对值、相对值（如 10px 或 2em）或百分数。

(3) 如果愿意，重复第 (2) 步，添加其他方向上的偏移量。每个属性 / 值对之间用分号分隔。

提示　相对定位中的“相对”指的是相对于元素的原始位置，而不是周围元素的位置。你无法让元素相对于其他元素移动，但可以相对于它的原始位置移动。这一点很重要！

提示　其他元素不会受到偏移的影响，它们仍然按照这个元素原来的盒子进行排列。设置了相对定位的内容可能与其他内容重叠，这取决于 top、right、bottom 和 left 的值。

提示　使用相对定位、绝对定位或固定定位时，对于相互重叠的元素，可以用 z-index 属性指定它们的叠放次序。详细信息参见 11.15 节。

提示　如果不指定 position 属性，偏移量将不会起作用。

提 示　对元素设置 position: static，可以覆盖 position: relative 设置。static 是元素的默认值，这就是元素出现在常规文档流中的原因。有关示例参见 11.15 节。

提 示　定位不是继承的。

11.14　对元素进行绝对定位

正如前面提到的，网页中的元素通常按照它们在 HTML 源代码中出现的次序进行排列（如图 11.14.1 所示）。也就是说，如果 img 元素出现在 p 元素之前，图像就出现在段落的前面。通过对元素进行绝对定位，即指定它们相对于 body （参见图 11.14.2 和图 11.14.3）或最近的已定位祖先元素（参见图 11.14.4）的精确位置，可以让元素脱离正常的文档流。

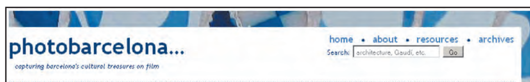


图 11.14.1　根据常规的文档流次序，搜索框仍然位于主导航的下方。我们希望将它移至包含它的报头的右上角

```
#masthead form {
     position: absolute;
     top: 7px;
     right: 0;
}
```

图 11.14.2　通过对搜索框进行绝对定位，让它完全脱离了文档流。它不知道其他内容的存在，其他内容也不知道它的存在。仅用这段代码还不能实现目的，因为在没有另外进行指定的情况下，设置 position: absolute 的元素是相对于 body 进行定位的，如图 11.14.3 所示



图 11.14.3　搜索框显示在距离 body 上边缘 7 像素、右边缘 0 像素的位置

```
/* 这个div既包围搜索表单，又包围主导航 */
#masthead div {
     float: right;
     position: relative;
}

#masthead form {
     position: absolute;
     top: 7px;
     right: 0;
}
```

图 11.14.4　为搜索框的 div 容器设置 position: relative;，从而让搜索框相对于该 div（而不是 body 元素）进行绝对定位。这可以让搜索框处在我们希望的位置，但同时也引入了另外一个问题（参见图 11.14.5）

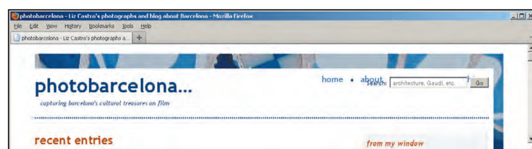


图 11.14.5　现在，搜索框显示在距离其容器 div 上边缘 7 像素、右边缘 0 像素的位置（搜索框上方和右侧还有为 #page 设置内边距产生的宽 10 像素的留白）。不过，搜索框显示在导航的上面，这不太好。前面提到，对元素进行绝对定位以后，它就会脱离文档流，因此导航仍显示在它本来的位置，就像搜索框并不存在。随后我们将解决这个问题（参见图 11.14.6）

对元素进行绝对定位的步骤

(1) 输入 position: absolute;（不要忘了输入分号；，空格是可选的）。

(2) 根据需要，输入 top、right、bottom 或 left。

再输入：*v*，这里的 *v* 是希望元素相对于其祖先元素偏移的距离（如 10px 或 2em）或相对于其祖先的百分比（有关解释参见第二条提示）。

(3) 根据需要，对其他方向重复第 (2) 步，照例用分号分隔每个属性 / 值对。

(4) 根据需要，对希望成为绝对定位参照体的祖先元素添加 position: relative（如图 11.14.4 所示）。如果跳过这一步（如图 11.14.2 所示），元素将相对于 body 计算偏移量（如图 11.14.3 所示）。

提 示　由于绝对定位的元素脱离了文档流，因此它们可能会相互重叠，或与其他元素重叠（参见图 11.14.6）（这不一定是坏事儿）。

```
/* 这个div既包围搜索表单，又包围主导航 */
#masthead div {
     float: right;
     position: relative;
}

#masthead form {
     position: absolute;
     top: 7px;
     right: 0;
}

.nav {
     margin-top: 45px;
}
```

图 11.14.6　仅为导航设置一个上外边距就可以将它推到搜索框的下方，让并它与标志中的标语对齐（如图 11.14.7 所示）

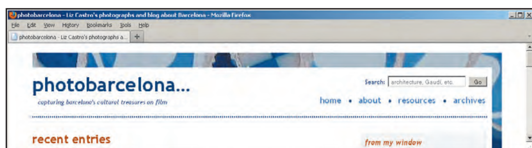


图 11.14.7　现在，搜索框和导航显示为我们希望看到的样子。更好的是，当页面变窄时，它们仍保持相对位置不变（参见图 11.14.8）。这对使用移动电话和其他窄屏幕设备的访问者来说是个好消息

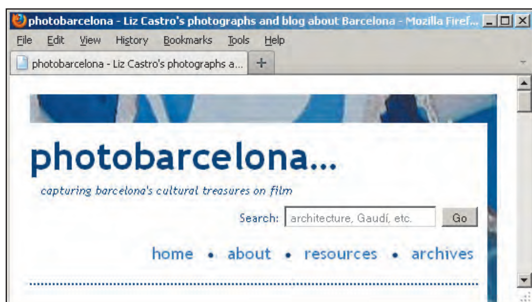


图 11.14.8　由于搜索框和导航都包含在同一个 div 中，因此它们是作为一个整体进行移动的。当浏览器窗口变窄时，它们在浮动定位的标志下方进行滑动。由于布局需要考虑这些情形，因此这种体验很实用

提 示　如果不为绝对定位的元素指定偏移量，这个元素将显示在它的自然位置上，但不会影响后续元素在文档流中的位置。

提 示　还有一种定位类型称为固定定位。当访问者滚动浏览器窗口时，页面内容通常随之上下移动。如果对元素设置 position: fixed;，它就会固定在浏览器窗口中。当访问者上下滚动浏览器窗口时，该元素不会随之移动，页面的其余部分仍照常滚动。IE6 不支持 fixed。

提 示　使用相对定位、绝对定位或固定定位时，可以使用 z-index 属性指定相互重叠的元素的叠放次序。详细说明参见 11.15 节。

提 示 对元素设置 position: static 将覆盖 position: absolute; 的设置。static 是元素定位的默认值，这也是元素为什么出现在常规文档流中的原因。有关实例参见 11.15 节。

提 示 定位不是继承的。

11.15 指定元素的三维位置

当你开始使用相对定位、绝对定位和固定定位以后，很可能发现元素相互重叠的情况，就像上面出现过的搜索框和主导航一样。你可以选择哪些元素应该出现在顶层（参见图 11.15.1、图 11.15.2 和图 11.15.3）。

```
...
<body>
<div class="box1">
    <p>This is box 1</p>
</div>

<div class="box2">
    <p>This is box 2</p>
</div>

<div class="box3">
    <p>This is box 3</p>
</div>

<div class="box4">
    <p>This is box 4</p>
</div>
</body>
</html>
```

图 11.15.1 这是一段很简单的 HTML 代码，其样式表为图 11.15.2，显示效果如图 11.15.3 所示

指定元素三维位置的方法

输入 z-index: *n*，其中的 *n* 是表示元素在定位过的对象堆中的层级的数字。

```
div {
    background: #ccc;
    border: 1px solid #666;
    height: 125px;
    position: absolute;
    width: 200px;
}

.box1 {
    background: pink;
    left: 110px;
    top: 50px;
    z-index: 120;
}

.box2 {
    background: yellow;
    left: 0;
    top: 130px;
    z-index: 530;
}


.box3 {
    height: auto;
    min-height: 125px;
    position: static;

    /* 对叠放顺序的设置没有效果，因为元素的
     → 定位方式不是绝对定位、相对定位和固定
     → 定位中的一种*/
    z-index: 1000;
}

.box4 {
    background: orange;
    left: 285px;
    top: 65px;
    z-index: 3;
}
```

图 11.15.2 对于使用绝对定位（或相对定位、固定定位）的元素，z-index 最高的数显示在最上面（如图 11.15.3 所示），不管该元素在 HTML 中出现的次序（如图 11.15.1 所示）。这也显示了 position: static; 的方便之处。第一条规则为所有四个 div 设置了 position: absolute;，而定义 .box3 时又覆盖了这一规则，让 .box3 回到默认的 static。这让 .box3 回归常规的文档流，因此，尽管它的 z-index 值最高，但这没有任何效果，它总是位于最底层

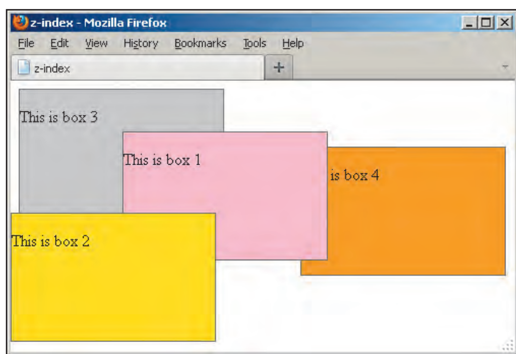


图 11.15.3 定位的盒子按照 z-index 由高到低的次序进行叠放。第三个盒子位于最底层，因为它处于常规的文档流

提示 z-index 属性仅对定位过的元素（即设为绝对定位、相对定位或固定定位的元素）有效。图 11.15.1 仅包含绝对定位的元素，但实际上可以对绝对定位、相对定位和固定定位的元素混合使用 z-index，z-index 会将它们作为整体进行安排，而不是分别安排。

提示 z-index 的值越大，元素在堆中就越高（参见图 11.15.1 和图 11.15.2）。

提示 如果在设置了 z-index 的元素中嵌套元素，那么这些嵌套的元素首先根据其自身的 z-index 确定次序，然后作为一个整体在更大的上下文中确定次序。

提示 IE7 及之前的版本无法正常处理 z-index，每个定位的元素都会生成自己的排序上下文，而不是相对于整个页面所有定位元素形成的堆进行排序。该问题的描述及解决方案参见 http://brenelz.com/blog/squish-the-internet-explorer-z-index-bug/。（该方案使用了内联样式，请忽略这一点。应该将 CSS 放到外部样式表中。）

提示 z-index 属性不是继承的。

11.16 决定溢出的位置

元素并不总是包含在它们自己的盒子里。这可能是因为盒子不够大，例如，图像比它的容器更宽就会溢出；也可能是因为使用负的外边距或绝对定位让内容处于盒子的外面。无论是哪种情况，都可以使用 overflow 属性控制元素在盒子外面的部分。

决定浏览器如何处理溢出的步骤

(1) 输入 overflow:。

(2) 输入 visible，让元素的盒子进行扩展以适应其内容。这是默认项（参见图 11.16.1）；

或者输入 hidden，隐藏元素的盒子容纳不了的内容（参见图 11.16.2 和图 11.16.3）；

或者输入 scroll，从而总是在元素上添加滚动条，让访问者可以通过滚动条看到溢出的内容（参见图 11.16.4 和图 11.16.5）；

或者输入 auto，让滚动条仅在需要的时候出现。

提示 实践中我并不赞成像示例那样隐藏图像，因为让用户可以看到它们更合适，不管浏览器的宽度是多少（记住，移动电话和平板电脑的屏幕较窄）。由于示例中的修改仅作演示用，因此我在本书网站上的完整代码中去掉了 height 和 overflow 声明。同时，我不会再像示例那样在所有页面引入这么多图像，因为加载这些图像是很大的负担。在示例中这样做只是出于演示的目的。

提示 overflow 属性还可以很方便地清除 float。参见 11.11 节的“清除 float 的其他方法”。

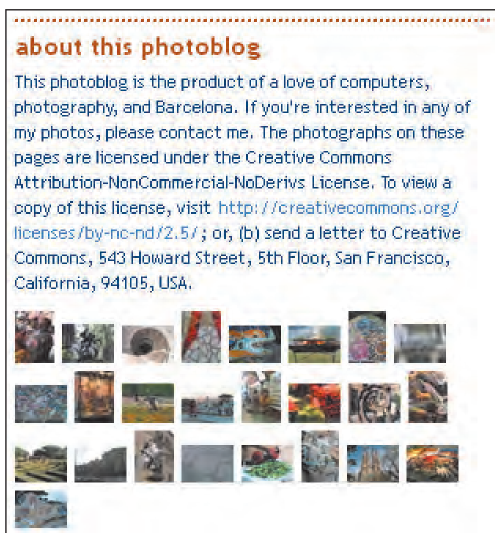


图 11.16.1　当窗口变窄时，页脚下面的图像会以多行显示。这通常是好事，因为我们希望内容可以适应不同的环境。不过，为了演示 overflow 属性的处理机制，我们将临时地修改这一行为（如图 11.16.2 和图 11.16.3 所示）

```
.thumbnails {
    height: 33px;
    overflow: hidden;
}
```

图 11.16.2　为了总是在一行之内显示图像（不管浏览器的宽度），我们将 ul 元素的高度设为这些图像中高度最大的图像的高度，并设置 overflow 属性为 hidden

提 示　注意，当子元素比父元素大的时候，IE6 会错误地将父元素扩展到跟子元素一样大。唯一的例外是将 overflow 属性设为 visible（默认值）以外的值，在这种情况下父元素会变回它的正常大小，并让 overflow 属性发挥作用。

提 示　overflow 属性的默认值是 visible。overflow 属性不是继承的。



图 11.16.3　现在，超出范围的图像被隐藏了。如果扩大浏览器窗口，这一行就会显示更多的图像。下面将演示另一种方法（参见图 11.16.4），在这个例子中效果并不好（如图 11.16.5 所示）

```
.thumbnails {
    height: 33px;
    overflow: auto;
}
```

图 11.16.4　如果要让图像的可视区域仍限制在一行以内，同时让访问者可以通过滚动条看到显示为多行的图像（如图 11.16.5 所示），可以使用 overflow: auto;，同时结合前面设置的高度



图 11.16.5　这是将滚动条拖到最后一行附近的样式。显然，这种效果用在这里并不合适。尽管为容器设置了更大的高度，但这种技术在某些情形下可能很方便

11.17　垂直对齐元素

可以使用除默认方式以外的多种方式对齐元素，让它们在页面上显得较为整齐（如图 11.17.1、图 11.17.2 和图 11.17.3 所示）。

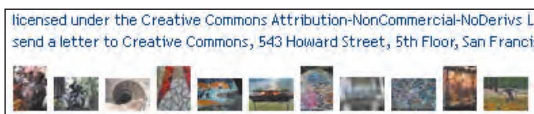


图 11.17.1　图像默认对齐行的底部

```
.thumbnails img {
    vertical-align: middle;
}
```

图 11.17.2　注意，对齐是对图像本身设置的，而不是对包含图像的列表项目（li）设置的。（关于列表，参见第 15 章。）

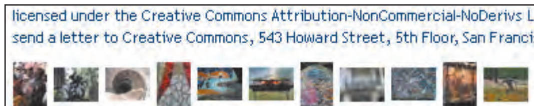


图 11.17.3　现在，图像对齐行的中线

使元素垂直对齐的步骤

(1) 输入 vertical-align:。

(2) 输入 baseline，使元素的基准线对齐父元素的基准线；

或者输入 middle，使元素的中线对齐父元素的中线；

或者输入 sub，使元素成为父元素的下标；

或者输入 super，使元素成为父元素的上标；

或者输入 text-top，使元素的顶部对齐父元素的顶部；

或者输入 text-bottom，使元素的底部对齐父元素的底部；

或者输入 top，使元素的顶部对齐当前行里最高元素的顶部；

或者输入 bottom，使元素的底部对齐当前行里最低元素的底部；

或者输入元素行高的百分比，可以是正数，也可以是负数。

提示　vertical-align 属性仅对显示为行内元素的元素有效，对显示为块级元素的元素无效。更多细节参见 Chris Coyier 的解释：http://css-tricks.com/what-is-vertical-align/。

11.18　修改鼠标指针

一般情况下，浏览器负责控制鼠标指针的形状。大多数时候使用箭头形状，指向链接时使用手指形状（参见图 11.18.1），等等。使用 CSS 可以修改指针的形状（如图 11.18.2 和图 11.18.3 所示）。

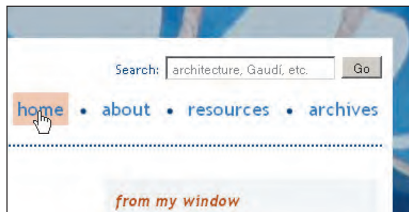


图 11.18.1　指向 Home 链接时，指针变成手指形状，同时链接高亮显示。对于其他链接也是这样

```
a.current {
    color: #1d3d76;
}

a:hover.current {
    background: white;
    cursor: default;
}
```

图 11.18.2　当访问者位于主页时，我为 home 链接添加了 class="current"。这样就修改了 home 链接的鼠标停留状态的颜色、指针和背景颜色，让它看起来不像是链接（另外，在这个例子中，也可以在导航中去掉 home 链接周围的 a 元素）



图 11.18.3 尽管它仍然是真正的链接，但它看起来已不像是链接。访问者正位于这个链接指向的页面，因此这样做是有意义的

修改指针形状的步骤

(1) 输入 cursor:。

(2) 输入 pointer 表示停留在链接上时通常显示的指针形状（）或 default 表示箭头（），或者输入 crosshair（+）、move（）、wait（）、help（）、text（I）或 progress（）；

或者输入 auto 显示特定情形下通常使用的指针形状；

或者输入 *x*-resize 显示双向箭头，这里的 *x* 是其中一个箭头需要指向的方向，可以是 n（北）、nw（西北）、e（东），等等。例如，e-resize 指针显示成 ↔。

提 示 不同浏览器、不同系统的指针形状可能有细微的差异。

11.19 显示和隐藏元素

图 11.19.1 中的示例页面可以演示 display 和 visibility 属性的差异。

display 属性可以用在多个方面。可以覆盖元素的自然显示方式，如将其从 inline 改为 block（参见图 11.19.2、图 11.19.3 和图 11.19.4），或者反过来。还有一种混合显示方式称做 inline-block，让元素与其他内容出现在同一行，同时具有块级元素的属性。display 属性还可以让元素及其内容不在页面上占据任何空间（如图 11.19.5 和图 11.19.6 所示）。此外，该属性还可以是其他的值（参见提示）。

```
…
<body>
    <img src="assets/img/top.jpg" width="300"
    → height="125" alt="At the top" />
    <img src="assets/img/middle.jpg"
    → width="300" height="100" alt="In the
    → middle" class="hide" />
    <img src="assets/img/bottom.jpg"
    → width="300" height="125" alt="At the
    → bottom" />
</body>
</html>
```

图 11.19.1 这是 HTML：三个简单的 img 元素。中间的 img 为 hide 类，这在后面的示例中会用到。默认情况下，img 元素显示为行内元素（参见图 11.19.2）

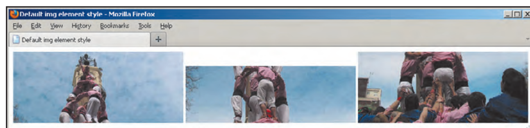


图 11.19.2 没有应用 CSS 时，图像挨着显示，因为 img 元素拥有 display: inline 的默认样式，就像短语内容元素一样（如果浏览器窗口变窄，图像就会换行以适应新的宽度）。很容易通过修改样式让它们分别位于单独的行（如图 11.19.3 所示）

```
img {
    /* 让元素显示在单独的一行上 */
    display: block;
}
```

图 11.19.3 为图像设置 display: block;，它们就会显示在单独的行，就像段落等默认显示为块级元素的元素那样（参见图 11.19.4）

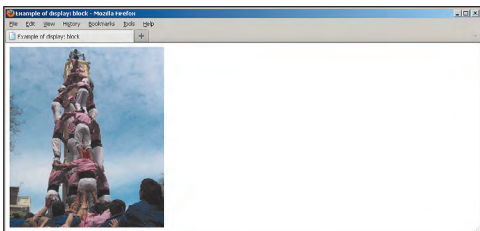


图 11.19.4　看起来就像是一个图像，但实际上它仍是来自图 11.19.1 的示例代码的三个 img 元素。唯一的区别是它们均根据图 11.19.3 中的样式规则显示为块级元素，而不是行内内容。我让浏览器保持相同的宽度，从而可以看到图像显示在各自的行仅仅是由于修改其 display 属性的原因

```
img {
    display: block;
}

.hide {
    /* 让这个类的所有元素都不显示 */
    display: none;
}
```

图 11.19.5　你应该记得，第二个 img 有一个 hide 类（参见图 11.19.1）。当我们设置 hide 为不显示时……



图 11.19.6　……第二个图像没有留下任何痕迹（我让浏览器窗口变窄了一些，但这不会影响显示的结果）

同时，visibility 属性的主要目的是控制元素是否可见。与 display 属性不同的是，使用 visibility 隐藏元素时，元素及其内容应该出现的位置会留下一片空白区域（如图 11.19.7 和图 11.19.8 所示）。

1. 指定元素显示方式的步骤

(1) 在样式表规则中，输入 display:。

(2) 输入 block，让元素显示为块级元素（就像开始新的段落），参见图 11.19.2、图 11.19.3 和图 11.19.4；

或者输入 inline，让元素显示为行内元素（不同于开始新的段落）；

或者输入 inline-block，让元素显示为行内元素，同时具有块级元素的特征，即可以为元素设置 width、height、margin、padding 等属性；

或者输入 none，隐藏元素，并将其从文档流中完全移除（参见图 11.19.5 和图 11.19.6）。

关于 display 属性的其他值，参见提示。

2. 控制元素可见性的步骤

(1) 在样式表规则中，输入 visibility:。

(2) 输入 hidden，让元素不可见，但在文档流中保留它(如图 11.19.7 和图 11.19.8 所示)；

或者输入 visible，让元素显示出来。

提 示　如果使用 display: none;，那么隐藏的元素不会在浏览器中留下任何痕迹。没有空白的空间（参见图 11.19.6）。如果使用 visibility: hidden;，隐藏的元素仍然会在文档流中保留其所占据的空间（参见图 11.19.8）。所有的内容（包括后代元素）也受到同样的影响。例如，如果对一个包含几个 p、figure 及 img 元素的 article 元素设置 display: none;，那么这些元素都不会显示。如果对该 article 元素设置 visibility: hidden;，就会留下一片空白的空间（可能很大！）。

提 示　对一个默认具有 display: inline; 样式的元素设置 display: block; 的例子见 11.11 节中的示例（图 11.11.3）。在那个例子中，我将其应用到了 span 元素中，span 元素包住了网站标志的标语。

提 示　display 还有其他几个值，尽管 IE6 和 IE7 对其中的一些并不支持。更多信息参见 http://reference.sitepoint.com/css/display（一定要阅读评论）。

提 示　visibility 属性不必如图 11.19.7 中那样与 display 属性结合使用，反之亦然。

提 示　visibility 属性还有第三个值（如果不算 inherit 的话）：collapse，用于 table 元素的部分内容。IE6 和 IE7 并不支持这个值。关于 collapse 的更多信息，参见 http://reference.sitepoint.com/css/visibility。

```
img {
    display: block;
}

.hide {
    /* 隐藏这个类的所有元素 */
    visibility: hidden;
}
```

图 11.19.7　在 hide 类中移除 display: none; 声明，并将 visibility 属性的值改为 hidden……

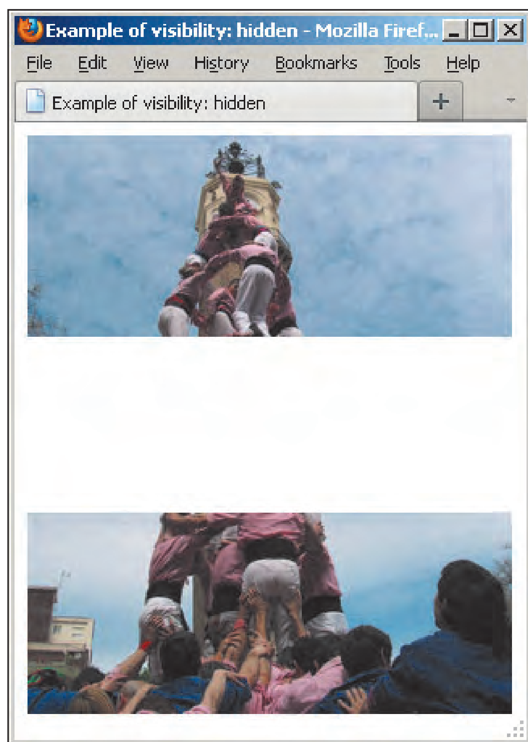


图 11.19.8　图像隐藏后，图像原先的位置呈现空白

第 12 章

12

样式表——从移动设备到台式机

本章内容

- ❑ 移动战略和注意事项
- ❑ 理解和实现媒体查询
- ❑ 构建适用于媒体查询的页面

最后一分钟决定要不要去看电影；就安道尔的官方语言跟人打赌；查找已经迟到 15 分钟的会议所在公司的电话号码；查看去这家公司的地图（因为迟到的原因是找不到这家公司）……

我们想要及时地获取信息，而随着各种形状和尺寸的移动设备的剧增，万维网可以出现在你的口袋里、钱包里、背包里，就像它在书桌或餐桌上一样方便。随着苹果公司的 Mobile Safari 浏览器的出现以及 iPhone 的流行，如今的移动浏览器同几年前相比已有巨大的不同。

现在，网站的建设者需要让访问者能够通过移动电话、智能手机、平板电脑、笔记本、台式机、游戏机以及未来任何可以上网的设备获取信息。

在本章中，你将了解到如何构建在各种设备上都能正常工作的网站，它能根据设备功能的不同对布局进行调整。

12.1　移动战略和注意事项

创建适用于移动设备的网站通常有两种方法。

- ❑ 建立专门用于移动电话的站点，即专门为移动体验打造，且与为桌面计算机和平板电脑用户准备的网站独立的网站。有时还为平板电脑（尤其是 iPad）建立单独的网站，因此总共至少有三个网站。
- ❑ 构建一个适用于所有情况的网站。为所有的设备（从移动电话到桌面计算机）准备同样的 HTML，再分别设置适用于不同设备的样式。通过一些额外的手段，甚至可以为移动电话用户准备不同的图像和视频尺寸，让他们不必因下载太大的文件而遇到麻烦。

（注意，我在本章中常使用“桌面计算机”一词同时指代台式机和笔记本。）

没有一种正确的方法可以适应所有的情形。不过，科技与 Web 开发技术的最新发展已将单一站点方法推向了 Web 社区讨论的前沿。稍后即会解释这一点。

1. 专门用于移动电话的网站

这种方法基于这样的思想：用于桌面计算机的网页和用于移动电话的网页之间的差异并不仅仅体现在显示尺寸上，更重要的是环境不一样。这种方法认为，访问者在旅途中需要用移动电话获取的信息与他们在家里或办公室里要查找的信息很不一样，因此单纯地将来自桌面计算机网站的信息小型化是不够的。同时，与移动电话相比，桌面计算机变得越来越强大，互联网连接速率越来越快，因此它们有机会提供更丰富的体验。

这种方法要求决定哪些信息是对移动用户特别有用的，并为他们提供专门针对移动设备的网站，使用户可以通过最少的滚动、触碰、点击、下载和等待时间使用网站。

尽管这种方法并非只针对大型公司和公共服务网站，但实际示例大都来自这些网站。对此，很大一部分原因是这些网站更容易承担开发和维护多站点的任务。Amazon、Target 等购物网站通常极大地简化其移动站点的首页，并修改导航的策略。例如，Target（www.target.com）考虑到移动电话访问者很有可能是在户外访问网站，并查找附近的实体店，因此它在其移动站点上突出了商铺位置的信息。同时，该网站为小屏幕用户削减了商品分类数量，为他们提供了完全不同的导航商品分类的方式。

旧金山湾区捷运系统（Bay Area Rapid Transit，BART）是一家提供公共运输服务的机构，受众广泛。它们为访问者提供三种版本的网站：桌面站（www.bart.gov，如图 12.1.1 所示）、移动站（m.bart.gov，如图 12.1.2 所示）和为旧设备提供的更为精简的移动版（www.bart.gov/wireless/）。同时，他们允许用户通过页脚的链接（参见图 12.1.2）在不同版本之间进行切换，将控制权交到访问者手里。



图 12.1.1　这是在桌面计算机上访问 BART 网站的样子。对于其他一些设备，如 iPad，默认也会显示这个版本。页脚提供了切换到移动版的链接（可参见图 12.1.2），此处没有显示出来。

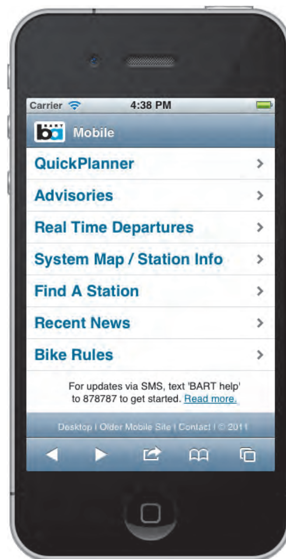


图 12.1.2　BART 的移动版网站精简了界面，省略了桌面站（参见图 12.1.1）上的图片，让访问者可以更快地找到信息。如果查看页脚，会发现 Desktop（桌面版）和 Older Mobile Site（旧移动版）两个链接，前者指向如图 12.1.1 所示的网站，后者指向一个更为精简的移动版网站（www.bart.gov/wireless/）

耐克公司的网站也提供了至少三种版本——用于智能手机的版本、用于 iPad 的版本以及用于桌面计算机的版本。它们包含的图像依次变得丰富。类似地，雅虎网站也有移动设备、平板电脑和桌面计算机的体验。

2. 一个网站适应所有情形

看起来，每周都有新的设备投放到市场上，毫无疑问，各家公司都在设计新的设备类型。构建和维护多个网站现实吗？甚至说，有必要吗？我们无法预知未来的情形，因此这种方法主张构建适用于所有设备的网站，然后根据不同的设备调整布局。

对大多数人来说，单一网站就能满足访问者的需求。此外，如今预测访问者访问网站的目的已变得更为困难，特别是在智能手机及其浏览器变得更为强大的情况下，这也是采用单一网站的原因之一。例如，移动用户并不一定处于移动的状态。下面这种情形一定发生在很多人身上：闲坐在沙发上用手机上网，尽管笔记本电脑就在房间的另一侧。大多数情况下，我想看到完整的站点，而不是桌面体验的移动化版本。

不过，我们无法绕开的事实是移动电话的屏幕尺寸和带宽更小，因此我们仍需花工夫创建适应这种环境的网站。

3. 一个网站适应所有情形：着手实现

一个网站的方案听起来的确不错，不过怎样才能适应如此繁多的设备呢？

这是渐进增强值得称道的地方（请回顾前言中的“渐进增强：一种最佳实践”一节）。由于 HTML 与 CSS 是分离的，因此可以按照分辨率逐渐增高、设备功能逐渐增强的次序，提供逐渐增强的布局（如图 12.1.3、图 12.1.4 和图 12.1.5 所示）。



图 12.1.3　不管你信不信，这里以及图 12.1.4 和图 12.1.5 显示的 Food Sense 主页都来自同一个网站（www.foodsense.is），而不是各自具有单独 URL 的不同网站。这个网站采用了响应式 Web 设计方法，因此它的布局可以根据不同的查看环境进行改变。iPhone（如本图所示）及其他屏幕大小相似的设备会根据特定的 CSS 规则显示布局。对于更大的浏览器窗口（参见图 12.1.4 和图 12.1.5），则有另外一些 CSS 规则控制相应的布局

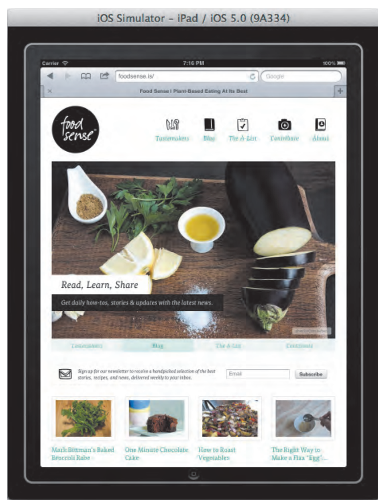


图 12.1.4　这是使用 iPad 及其他屏幕大小相似的设备查看 Food Sense 主页的样子。由于浏览器有更多空间显示内容，因此这套样式的 CSS 对标识和导航都进行了调整

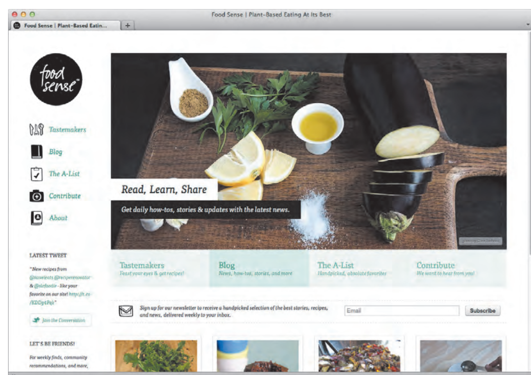


图 12.1.5 这是 Food Sense 主页显示在桌面浏览器上的样子，也是最宽的显示效果。该站还有另外两个布局没有展示出来。你可以在计算机上访问 www.foodsense.is，再拖动浏览器的一角让浏览器窗口变窄或变宽并查看效果

Ethan Marcotte 为我们提供了如何实现上述效果的蓝图，并将相应的方法命名为响应式 Web 设计（responsive Web design）。他的文章（www.alistapart.com/articles/responsive-web-design/）和 *Responsive Web Design*（A Book Apart，2011）一书均获得了极高的评价。他的方法植根于以下三点。

- 灵活的、基于网格的布局。这就是你在第 11 章看到的流式布局（有些微调整）。对于响应式站点，所有的 `width`、`margin` 和 `padding` 属性都用百分数设定，因此所有的布局成分都是相对的。
- 灵活的图像和媒体。图像和媒体资源的尺寸也是用百分数定义的，从而可以根据环境进行缩放。（参见 Ethan 的书摘：www.alistapart.com/articles/fluid-images/。）随着技术的发展，现在已经可以根据不同的设备屏幕尺寸呈现不同尺寸的图像，从而让使用移动电话的用户不必等待大尺寸图像的下载。
- 媒体查询。媒体查询是 CSS3 的组件之一，使用这项技术，可以根据媒体特征（如浏览器可视页面区域的宽度）对设计进行调整（参见下一节的“理解视觉区域及使用视觉区域 `meta` 元素”）。你将在 12.2 节了解更多的信息，并在 12.3 节看到有关的实例。

Web 社区围绕响应式站点的讨论已经越来越多了，并分享了大量基于 Ethan 方法的技术。这种方法并不仅仅是博客上的讨论。《波士顿环球报》（www.bostonglobe.com）的新网站是基于响应式 Web 设计技术制作的，该站一经推出就成了热议的话题。

我们将在本章余下的部分重点介绍这种方法。从第 11 章起，你将了解到如何对网站运用移动优先的方法，使用媒体查询逐渐为更大的屏幕分辨率添加 CSS（参见图 12.1.3、图 12.1.4 和图 12.1.5）。

不过，这并不是一刀切的解决方案。如前所述，访问环境以及所需的内容、导航、外观、交互的不同有可能要求建立分离的站点。

提 示 Luke Wroblewski 于 2009 年 11 月开始倡导“移动优先”的设计（www.lukew.com/ff/entry.asp?933）。其前提是在考虑移动体验的情况下设计网站，再针对桌面环境实现对应的组件（如果这些站点不一样的话）。根据他的观点，这样做更容易判断哪些内容对所有设备的用户都是最重要的。他针对该主题的演讲见 www.lukew.com/ff/entry.asp?1137。他还写了一本书，书名正是 *Mobile First*（A Book Apart，2011）。

提 示 Jeremy Keith 在题为“One Web”的演讲中归纳了“一个网站适应所有情形“的方法（www.vimeo.com/27484362/）。（如果你更倾向于阅读，可以参见相应的讲稿全文：www.adactio.com/articles/4938/。）

提 示 上述两个材料都值得高度推荐。其实，现在就可以观看了，我等着你！

12.2　理解和实现媒体查询

我们在 8.6 节中学习过，可以使用两种方式针对特定的媒体类型定位 CSS。（还有第三种方式，即使用 `@import` 规则，我们不讨论这种方法，因为它会影响性能。）回顾一下，第一种方式是使用 `link` 元素的 `media` 属性，例如 `<link rel="stylesheet" href="global.css" media="screen" />`（位于 head 内）。第二种方式是在样式表中使用 `@media` 规则，例如：

```
/* 打印样式表 */
@media print {
    header[role="banner"] nav,
    .ad {
        display: none;
    }
}
```

媒体查询增强了媒体类型方法，允许根据特定的设备特性定位样式（参见图 12.2.1）。要调整网站的呈现样式，让其适应不同的屏幕尺寸，采用媒体查询特别方便。下面列出了可以包含在媒体查询里的媒体特性。

- ❑ `width`（宽度）
- ❑ `height`（高度）
- ❑ `device-width`（设备宽度）
- ❑ `device-height`（设备高度）
- ❑ `orientation`（方向）
- ❑ `aspect-ratio`（高宽比）
- ❑ `device-aspect-ratio`（设备高宽比）
- ❑ `color`（颜色）
- ❑ `color-index`（颜色数）
- ❑ `monochrome`（单色）
- ❑ `resolution`（分辨率）
- ❑ `scan`（扫描）
- ❑ `grid`（栅格）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Media queries in link elements
    → </title>
    <meta name="viewport" content="width=
    → device-width, initial-scale=1.0" />
    <link rel="stylesheet" media="all"
    → href="base.css" />

    <!--
    逻辑是only
    类型是screen
    特性：min-width的值为480px
    -->
    <link rel="stylesheet" media="only
    → screen and (min-width: 480px)"
    → href="styles-480.css" />
</head>
<body>
...
```

图 12.2.1　base.css 中的样式用于所有的输出设备。styles-480.css 中的样式则仅用于支持媒体查询且视觉区域宽度不低于 480 像素的浏览器

还有一些非标准的媒体特性，如

- ❑ `-webkit-device-pixel-ratio`（WebKit[①] 设备像素比）

① WebKit 是 Chrome、Safari 等浏览器使用的网页引擎和 JavaScript 引擎开源程序。——译者注

- -moz-device-pixel-ratio（Mozilla[①] 设备像素比）

除了 orientation、scan 和 grid 以外，上述属性均可添加 min- 和 max- 前缀。min- 前缀定位的是“大于或等于”对应值的目标，而 max- 前缀定位的则是“小于或等于”对应值的目标。在本章里，我们将着重介绍 min-width 和 max-width，因为它们是制作响应式页面时反复用到的两个媒体特性。CSS3 媒体查询规范（www.w3.org/TR/css3-mediaqueries/#media1）中有对各项媒体特性的介绍。

现代桌面浏览器和移动电话浏览器对媒体查询的支持程度很高。不过，Internet Explorer 8 及以下版本并不支持媒体查询（针对这些浏览器的 min-width 和 max-width 解决方案见第一条提示）。

1. 媒体查询语法和示例

Peter Gasston 的 *The Book of CSS3*（No Starch Press，2011）一书写得很棒。该书对媒体查询的语法做了很好的归纳。以下是媒体查询的基本语法。

- 指向外部样式表的链接：

```
<link rel="stylesheet" media=
→"logic type and (feature:
→value)" href="your-stylesheet.
→css" />
```

- 位于样式表中的媒体查询：

```
@media logic type and (feature:
→value) {
  /* 目标CSS样式规则写在这里 */
}
```

```
/* 常规样式写在这里。每个设备都能获取它
→们，除非被媒体查询中的样式规则覆盖 */
body {
    font-size: 100%;
}

p {
    color: green;}

/*
逻辑是 only
类型是 screen
特性：min-width 的值为 480px
*/@media only screen and (min-width:
→480px) {
    /* 针对这种情形的样式写在这里 */
    p {
       color: red;
       font-weight: bold;
    }
}
```

图 12.2.2 这个粗略的示例包含了默认的段落样式，接着是当媒体查询结果为真时对段落文本的修改。我将这份样式表保存为 basic-media-query.css，并在如图 12.2.3 所示的页面中加载该文件。结果如图 12.2.4、图 12.2.5 和图 12.2.6 所示

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Basic media query example</title>
    <meta name="viewport" content="width=
    →device-width, initial-scale=1.0" />
    <link rel="stylesheet"
    href="assets/
    →css/basic-media-query.css" />
</head>
<body>
    <p>Hi, I'm a paragraph. By default, I'm
    →green and normal. But get me in a
    →viewport that's at least 480px wide,
    →and I get red and bold!</p>
</body>
</html>
```

图 12.2.3 这个页面包含指向外部样式表（参见图 12.2.2）的链接，该样式表包含一个基本的媒体查询示例

① Mozilla 是 Firefox 等浏览器的基础程序。——译者注

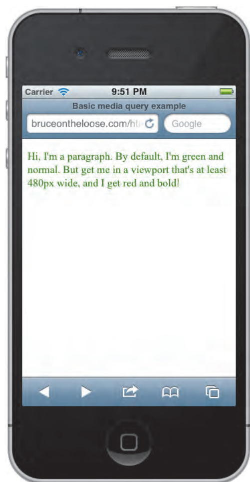


图 12.2.4　在 iPhone 纵向模式下，Mobile Safari 的视觉区域为 320 像素宽，因此文本仍为样式表中基准样式定义的绿色（它从浏览器的默认样式继承了 font-weight 的常规值）。不过，当页面显示在 iPad 中时……

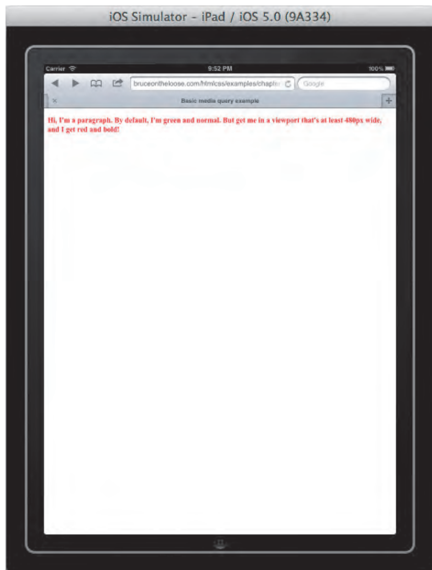


图 12.2.5　……由于在 iPad 纵向模式下浏览器的视觉区域为 768 像素宽，而宽度大于或等于 480 像素时会触发媒体查询，因此文本变成以红色、粗体显示。在 iPhone 横向模式下该效果也会生效，因为该模式下视觉区域的宽度刚好为 480 像素

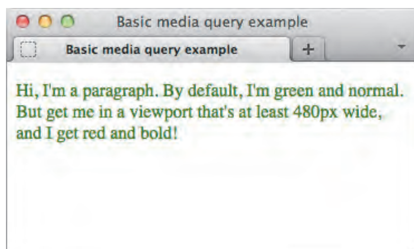


图 12.2.6　现代桌面浏览器也理解媒体查询。这是 Firefox，通过拖拽窗口的右下角使视觉区域的宽度小于 480 像素，因此文本显示为红色，font-weight 为常规值。如果让窗口变大，使其宽度至少为 480 像素，文字将立即以粗体、红色显示——无须刷新页面

我将在随后解释有关的语法，不过几个简单的例子（参见图 12.2.1 和图 12.2.2）有助于在上下文中理解这些语句。示例中的查询是相同的，但它们呈现样式的方式却是不同的。图 12.2.1 中的示例可以翻译为“仅当媒体类型为 screen 且视觉区域最小宽度为 480 像素时，加载并使用 styles-480.css 中的样式规则”。图 12.2.2 中的示例可以翻译为“仅当媒体类型为 screen 且视觉区域最小宽度为 480 像素时，使用下面的样式规则”。（关于视觉区域的含义，参见本节末尾的“理解视觉区域及使用视觉区域 meta 元素”。）我创建了一个测试页（如图 12.2.3 所示），并指向包含图 12.2.2 中代码的样式表。可以查看该页面在 iPhone（图 12.2.4）、iPad（图 12.2.5）及窄的桌面浏览器（图 12.2.6）中显示的样子。

回到语法，让我们看看这些代码的组成部分。

- *logic*（逻辑）部分是可选的，其值可以是 only 或 not。only 关键字可以确保旧的浏览器不读取余下的媒体查询，同时一并忽略链接的样式表。not 关键字可以对媒体查询的结果求反，让其反面为真。例如，使用 media="not screen" 会在媒体类型为 screen 以外

的任何类型时加载样式表。

- *type*（类型）部分是媒体类型，如 screen、print 等。
- *feature*: *value* 对是可选的，但一旦包含它们，它们必须用括号包围且前面要有 and 这个字。*feature* 是预定义的媒体特性，如 min-width、max-width、orientation 等。对 color、color-index 和 monochrome 特性来说，*value* 是可选的。

可以使用 and 将多个特性和值的组合串接起来，还可以创建一系列媒体查询（使用逗号分隔每个媒体查询）。在用逗号分隔的媒体查询列表中，如果有一个媒体查询为真，则整个媒体查询列表为真。图 12.2.7 和图 12.2.8 显示了多种媒体查询。

```
...
    <link rel="stylesheet" media="only
    → screen and (min-width: 480px)
    → and (max-width: 767px)" href=
    → "styles.css" />

    <link rel="stylesheet" media="only
    → screen and (orientation:
    → landscape)" href="styles.css" />

    <link rel="stylesheet" media="only
    → print and (color)" href="color-
    → pages.css" />

    <link rel="stylesheet" media="only
    → print and (monochrome)"
    → href="monochrome-pages.css" />

    <link rel="stylesheet" media="only
    → screen and (color), projection
    and
    → (color)" href="styles.css" />
</head>
<body>
...
```

图 12.2.7　当媒体查询为真时加载外部样式表的示例

```
/* 基准样式
------------------------------ */

/* 针对所有设备的基准样式 */


/* 开始媒体查询
------------------------------ */
@media only screen and (min-width:
480px)
→ and (max-width: 767px) {
    /* 样式规则 */
}

@media only screen and (orientation:
→ landscape) {
    /* 样式规则 */
}

@media only print and (color) {
    /* 样式规则 */
}

@media only print and (monochrome) {
    /* 样式规则 */
}

@media only screen and (color),
projection and (color) {
    /* 样式规则 */
}
```

图 12.2.8　这些媒体查询与图 12.2.7 中的是相同的，只是直接出现在样式表中

2. 指向外部样式表时定义媒体查询

(1) 在希望使用有关样式表的每个 HTML 页面的 head 部分，输入 <link rel="stylesheet"。

(2) 输入 media=" 以开始媒体查询。

(3) 根据“定义媒体查询的步骤”，创建媒体查询。

(4) 输入 " 以结束媒体查询。

(5) 输入一个空格，再输入 href="url.css"，这里的 url.css 是当媒体查询为真时应该应用到页面的样式表的路径和名称。

(6) 输入一个空格，最后输入 />。（如果

你愿意，也可以不输入空格，直接输入 >。这两种方式都是 HTML5 所允许的，其效果也是相同的。）

3. 在样式表内定义媒体查询并关联样式规则

(1) 在样式表内，输入 `@media` 和一个空格。

(2) 根据“定义媒体查询的步骤”，创建媒体查询。

(3) 输入一个空格和 `{`。

(4) 在新的一行（可选）创建当媒体查询为真时应该应用到页面的样式规则。

(5) 在新的一行（可选）输入 `}` 以结束媒体查询块。

4. 定义媒体查询的步骤

(1) 可选，输入 `only` 和一个空格。（尽管这一步是可选的，但我推荐包含 `only`，除非指定了 `not`。）如果不指定 `only`，输入 `not` 和一个空格（可选），表示条件为媒体查询的反面为真。

(2) 输入 `type`，这里的 `type` 是媒体类型（通常为 `screen` 或 `print`，参见 8.6 节）。

(3) 可选，输入一个空格、`and` 和另一个空格。再输入 (*feature: value*)，其中 *feature* 是某种预定义的媒体特性（包括 `width`、`height`、`device-width`、`device-height`、`orientation`、`aspect-ratio`、`device-aspect-ratio`、`color`、`color-index`、`monochrome`、`resolution`、`scan` 以及 `grid`），*value* 是合适的 *feature* 值（通常情况下以像素或 em 表示，但也并非总是如此，有关示例参见图 12.2.2、图 12.2.7 和图 12.2.8）。

(4) 如果想创建一系列媒体查询，输入一个逗号，再重复第 (2) 步和第 (3) 步。否则，媒体查询就定义完成了。

提示　要了解如何解决 IE8 及以下版本不支持媒体查询的问题，参见 12.3 节中的“在 IE8 及以下版本中呈现媒体查询样式。

提示　任何位于媒体查询以外的基准样式规则都会应用于所有的设备。可以使用媒体查询覆盖这些样式规则。需要说明的是，媒体查询样式规则声明仅在与常规样式**冲突**时进行覆盖，如图 12.2.5 中的 `color: green;`。如果媒体查询之前的 `p` 的样式规则包含 `font-style: italic;`，那么媒体查询为真时段落文本仍以斜体显示，因为媒体查询内的 `p` 的样式规则并未设置 `font-style`。

提示　iPhone 调高了横向模式下页面的缩放级别。因此，有些内容会显示到可视区域以外，需要访问者手动缩小页面，让宽度在屏幕边界以内。有一种方法可以防止这种现象的产生，但不幸的是，使用这种方法也会让访问者无法改变页面缩放级别。不过，如果你一定要控制这种行为，可以添加以下代码中突出显示的部分：`<meta name="viewport" content="width=device-width, initial-scale=1.0,` **`maximum-scale=1.0, user-scalable=no`**`"/>`。但我不推荐使用这种方法，应该忽略这两个属性，让访问者控制网站页面的缩放。

提示　可以使用苹果推出的免费的 iOS Simulator（iOS 模拟器）测试示例页面在 iPhone 和 iPad 中的显示情况。参见下一节的“移动编码和测试工具”。

理解视觉区域及使用视觉区域 meta 元素

视觉区域（viewport）指的是浏览器（包括桌面浏览器和移动浏览器）显示页面的区域。它不包含浏览器地址栏、按钮这样的东西，只是浏览区域。媒体查询的 width 特性对应的是视觉区域的宽度。不过，device-width 特性不是，它指的是屏幕的宽度。

在移动设备（如 iPhone）上，默认情况下这两个值通常不一样。Mobile Safari（iPhone 的浏览器）的视觉区域默认为 980 像素宽，但 iPhone 的屏幕只有 320 像素宽（高为 480 像素）。因此，iPhone 会像设为 980 像素宽的桌面浏览器那样显示页面，并将页面缩小以适应 320 像素的屏幕宽度（在纵向模式下），如图 12.2.9 所示。结果，当你在 Mobile Safari 中浏览大部分为桌面浏览器建立的网站时，会显示将这些网站缩小了的样子。在横向模式下也是这样处理的，只不过宽度为 480 像素。如图 12.2.9 所示，如果不进行放大，页面通常是难以阅读的（注意不同设备的默认视觉区域宽度并不相同）。

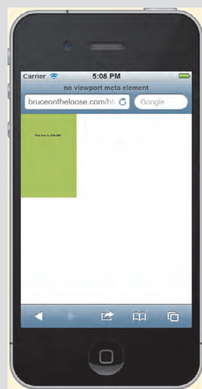


图 12.2.9　我的测试页面包含一个 320 像素 × 480 像素大小、绿色的 div。Mobile Safari 的视觉区域默认为 980 像素宽，因此 iPhone 会将该 div 缩小，以在 320 像素宽的屏幕内显示它。这就是这个绿色盒子大约占据屏幕宽度的三分之一（即 320/980）的缘故

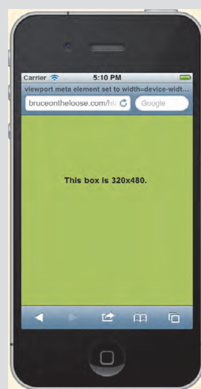


图 12.2.10　这个测试页面的代码与图 12.2.9 中页面的代码相比，除了包含设置了 width=device-width 的视觉区域 meta 元素以外，其他内容是完全相同的。如你所见，现在的视觉区域宽度与屏幕宽度是相同的

幸好，对于流式布局（即在 CSS 中使用百分数宽度建立的布局）有一种简单的解决方案。在页面的 head 部分添加视觉区域 meta 元素即可。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Fancy page title</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    ...
```

```
</head>
<body>
…
```

这段代码的重点是 `width=device-width`。有了这条语句，视觉区域的宽度会被设成与设备宽度（对 iPhone 来说为 320 像素）相同的值，因此在纵向模式下该宽度的页面内容会填充整个屏幕（如图 12.2.10 所示）。如果不包含这一语句，使用媒体查询的 `min-width` 和 `max-width` 特性将不会得到预期的结果。

代码中的 `initial-scale=1.0` 部分对 `width` 和 `device-width` 值没有影响，但通常会包含这一语句。它将页面的默认缩放级别设成了 100%。你也可以指定小于 `1.0` 或大于 `1.0` 的值。

还有其他三种属性（不过这里并未演示）。将 `minimum-scale` 设成大于 `0` 且不超过 `10.0` 的值，可以有效地控制页面的最小缩放级别。关于 `maximum-scale` 和 `user-scalable` 属性的信息，参见提示。

12.3　构建适用于媒体查询的页面

上一节解释了媒体查询是如何发挥作用的。现在你将看到如何将媒体查询应用到完整的页面，从而让页面的布局适应设备视觉区域的大小。这就是 Ethan Marcotte 的响应式 Web 设计使用的技术。不过，我不会像响应式设计那样让图像大小适应视觉区域（这不是强制的）。我将使用第 11 章中作为示例的页面开始以下的讲解。

我不会展示应用于每个媒体查询块内的全部样式规则，因为各个网站的这些样式规则都不一样。重要的是了解如何建立响应式网站，以及用于实现响应式网站的媒体查询类型。完整的页面和代码见 www.bruceontheloose.com/htmlcss/examples/。

1. 创建内容和 HTML

一切应该从可靠的、认真考虑过的内容开始。如果你试着用占位符文本（如没有任何含义的 lorem ipsum[①]）设计和构建你的网站，当你填入真正的内容以后，你可能会发现形式与内容并没有结合得很好。因此，应该尽可能地将内容采集工作提前，从而对设计和开发满足访问者（和你）需求的网站更有信心。

示例页面的底层 HTML 与第 11 章的页面的代码是相同的，除了以下三处例外。

- 我在 head 元素中添加了 **`<meta name="viewport" content="width=device-width, initial-scale=1.0"/>`**。关于这行代码的作用，参见“理解视觉区域及使用视觉区域 `meta` 元素”。如果你在实现一种灵活的布局，建议你在页面中包含此 `meta` 元素。
- 我移除了页面底部的小的缩略图（有 20 多个这样的图像）。要注意带宽（以及设备能力），这些图片的数量超过

① 在西方排版、设计领域常用一段以“lorem ipsum”开头的拉丁文作为占位符，以测试排版、设计的效果。这段文本是经过改造的、没有意义的拉丁文，常用“lorem ipsum”指代这段文本。——译者注

了所有设备默认加载的建议数量。如果我想来得花哨些，我可以写一些服务器端代码或 JavaScript 动态地为较大的屏幕加载这些图像。不过这已经超出了本书的范围。

- 我在页面底部添加了对 `respond.js` 的请求，从而让媒体查询对 IE8 及以下版本也有效。参见“在 IE8 及以下版本中呈现媒体查询样式”。

2. 选择设计实现方法

至少有两种方法可以实现响应式页面。这两种方法使用相同的 HTML，只是某些 CSS 不一样。下面是对这两种方法的概述。

方法 1：为所有的设备建立基准样式，再从小屏幕(移动)做起，逐渐覆盖大屏幕(桌面)

(1) 首先为所有的设备提供基准样式（如图 12.3.1 所示）。这通常包括基本的字体格式、颜色，可能还要对默认的外边距和内边距设置进行微调，但不包括元素的浮动或定位。内容将按照常规的文档流由上到下进行显示。网站的目标是在单列显示样式中是清晰的、中看的（如图 12.3.2 所示）。这样，网站对所有的设备(无论新旧，只要带有 Web 浏览器)都具有可访问性。在不同设备下，外观可能有差异，不过这是在预期之内的，完全可以接受。

(2) 从这种样式开始，使用媒体查询逐渐为更大的屏幕（或其他媒体特性，如 `orientation`）定义样式。大多数时候，`min-width` 和 `max-width` 媒体查询特性是最主要的工具（参见图 12.3.3 ~ 图 12.3.10）。

这种方法通常称为移动优先的响应式 Web 设计。

方法 2：为桌面环境构建网站，再处理不同的设备屏幕尺寸

(1) 先为网站的桌面版本添加样式（参见第 11 章）。

(2) 使用媒体查询为其他更小的屏幕尺寸覆盖样式。

第一种方法遵循渐进增强的原则，因此它在 Web 圈子里具有很大的吸引力。为了让你掌握这种方法，我将在示例中使用这种方法。如果你想尝试第二种方法，可以以第 11 章结束时的页面为起点，根据你在这里学到的知识，为更小的屏幕尺寸添加媒体查询。

```
/* 基准样式
----------------------------------- */
body {
    color: #1d3d76;
    font: 100% "Trebuchet MS", Verdana,
    ➝ sans-serif;
}

h1,
h2,
h3,
h4,
h5,
h6,
.logo {
    color: #b74e07;
    font-weight: bold;
}

h1 {
    font-size: 1.25em; /* 24px/16px */
    text-transform: lowercase;
}

.nav li {
    display: inline;
    font-size: .7em;
}

...
```

图 12.3.1 应用于所有设备的基准样式示例。这些样式规则与你在本章之前看到的其他代码是类似的，只是它们没有由媒体查询包围



图 12.3.2　iPhone 支持媒体查询，但我还未将媒体查询添加到样式表里。我只写好了基准样式，因此这些屏幕截图可以表示不支持媒体查询的浏览器呈现页面的样子。页面的布局是线性的，右侧图像出现在 recent entries 的下方。页脚在这些内容的下面，但这里没有显示出来

3. 逐步完善布局

好了，现在你已经把内容聚集到了一起，用语义化 HTML 对它们进行了标记，并决定使用方法 1 实现你的设计。图 12.3.1 ~ 图 12.3.10 显示了如何从适用于所有设备的基准样式开始，逐渐添加样式，直到拥有一个适合一系列视觉区域尺寸和设备的布局。

如果用上响应式 Web 设计的术语，这一过程可以表述为，使用媒体查询为页面中的每个断点（breakpoint）定义样式。断点即内容需作适当调整的宽度。在本例中，我为下列断点创建了样式规则。记住，对于每个最小宽度（没有对应的最大宽度），样式定位的是所有宽度大于该 `min-width` 值的设备，包括台式机。

- ❑ 320 像素的最小宽度（如图 12.3.3 和图 12.3.4 所示）。定位纵向模式下的 iPhone、iPod touch、各种 Android 以及其他移动电话。
- ❑ 480 像素的最小宽度（如图 12.3.5 和图 12.2.6 所示）。定位大一些的移动电话，如某些 HTC 机型，以及横向模式下的大量 320 像素设备（iPhone、iPod touch 及某些 Android 机型）。
- ❑ 600 像素的最小宽度（如图 12.3.7 和图 12.3.8 所示）。主要是为窄的桌面浏览器设置这些样式，不过，它们照例适用于任何显示为此最小宽度的设备。
- ❑ 600 像素的最小宽度和 767 像素的最大宽度之间（参见图 12.3.7 和图 12.3.8）。在此范围内，报头的布局被切断了（无论对桌面浏览器窗口大小怎样调整，它基本上是可以看见的），因此我创建了这部分样式对其进行了清理，填补了与最后的断点之间的差距。
- ❑ 768 像素的最小宽度（如图 12.3.9 和图 12.3.10 所示）。这适应于各种新旧浏览器，以及 iPad 和其他的平板电脑。

```
/* 基准样式
---------------------------------- */
...

/* 大于等于320px
---------------------------------- */
@media only screen and (min-width:
320px) {

    .photo {
       float: left;
    }

}
```

图 12.3.3　我为视觉区域至少有 320 像素宽的设备添加了一条样式规则，让博客文章里的文本包在图像周围（参见图 12.3.4）。我并未将这条样式规则包含在基准样式里，因为一些移动电话（甚至是智能手机）的屏幕更窄，会让图像旁边的文本由于显示空间太窄而变得难以阅读

12



图 12.3.4　由于使用了图 12.3.3 中定义的媒体查询，博客文章中的文本环绕在图像的周围。该样式对 iPhone 是有效的，因为其纵向模式下的视觉区域有 320 像素宽

```
/* 基准样式
----------------------------------- */
...

/* 大于等于320px
----------------------------------- */
@media only screen and (min-width: 320px) {
    ...
}

/* 大于等于480px
----------------------------------- */
@media only screen and (min-width:
480px) {

    .intro {
       margin: -.9% 0 0 110px;
    }

    .entry .date {
       margin: 0;
       text-align: right;
       position: relative;
       top: -1em;
    }

    #main .continued {
       margin-top: -1%;
       text-align: right;
    }

}
```

图 12.3.5　现在，样式表中有了定位视觉区域至少为 480 像素的设备的媒体查询。这样的设备包括屏幕更大的手机（如某些 Android 机型），以及横向模式下的 iPhone（参见图 12.3.6）

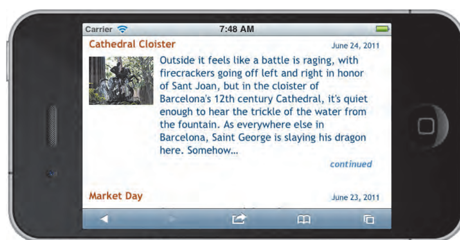


图 12.3.6　这是在 480 像素宽的屏幕下显示页面中部的样子。由于具有了更多的屏幕空间，我让文本不再包在图像周围，并让日期和 continued 向右对齐

```
/* 基准样式
----------------------------------- */
...

/* 大于等于320px
----------------------------------- */
@media only screen and (min-width: 320px) {
    ...
}

/* 大于等于480px
----------------------------------- */
@media only screen and (min-width: 480px) {
    ...
}

/* 大于等于600px
----------------------------------- */
@media only screen and (min-width:
600px) {

    #container {
       background: url(../img/bg-bluebench.
       ➝ jpg) repeat-y;
       margin: 20px auto;
       padding: 30px 10px 0 0;
       width: 90%;
    }

    .logo {
       float: left;
       font-size: 2em; /* 32px/16px */
    }

    ...
}

/* 600px～767px之间
----------------------------------- */
```

（接下页左栏代码）

（续上页右栏代码）

```
@media only screen and (min-width: 600px)
→ and (max-width: 767px) {

    .logo {
       background: #eee;
       font-size: 1.825em;
    }

    #masthead form {
       width: 235px;
    }

    input[type="text"] {
       width: 130px;
    }

    .nav li {
       font-size: .625em;
       font-weight: bold;
       padding-left: 1%;
    }

    ...
}
```

图 12.3.7　这些媒体查询开始让报头从线性布局转为水平样式。它让页面在宽度适中的桌面浏览器上显得更为好看（如图 12.3.8 所示）

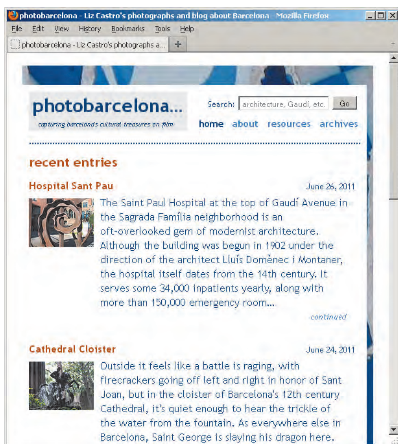


图 12.3.8　使用图 12.3.7 中的样式，页面已经接近其完整形式。现在，内容的布局仍为单栏，但搜索框和主导航移到了标识旁边。同时，页面周围的背景图像也第一次出现了

```
/* 基准样式
---------------------------------- */
...

/* 大于等于320px
---------------------------------- */
@media only screen and (min-width: 320px) {
    ...
}

/* 大于等于480px
---------------------------------- */
@media only screen and (min-width: 480px) {
    ...
}

/* 大于等于600px
---------------------------------- */
@media only screen and (min-width: 600px) {
    ...
}

/* 600px～767px之间
---------------------------------- */
@media only screen and (min-width: 600px) and
→ (max-width: 767px) {
    ...
}

/* 768px
---------------------------------- */
@media only screen and (min-width:
768px) {

    #container {
       max-width: 950px;
    }

    #page {
       padding-left: 0;
       width: 97.9167%;
    }

    .nav li {
       display: list-item;
       float: left;
       font-size: .75em; /* 12px/16px */
    }

    #main {
       float: left;
       width: 71%;
```

（接下页左栏代码）

（续上页右栏代码）

```
    }
    #related {
       margin-left: 72%;
    }

    #footer {
       clear: both;
    }

    ...
}
```

图 12.3.9　这是最终的媒体查询，定位至少有 768 像素宽的视觉区域。该媒体查询对大多数桌面浏览器（除非用户让窗口变窄，就像图 12.3.8 那样）来说都为真，它同时也适用于纵向模式下的 iPad 及其他一些平板电脑（参见图 12.3.10）



图 12.3.10　使用图 12.3.9 中的样式，页面变得完整了。这里显示的是在 iPad 中页面显示的样子，在桌面浏览器中（尽管要宽一些）也是类似的。由于宽度是用百分数定义的，因此主体内容栏和侧栏会自动伸展

你的断点可能跟此处用的不同。这取决于哪些断点对你的内容、设计和受众来说是合适的。

例如，有人会定义 `(min-width: 992px)` 的媒体查询，有时还会为更高的分辨率再定义一个媒体查询。相反，我在示例页面的 `#container` 选择器中添加了 `max-width: 950px;`（参见图 12.3.9）。这样，页面在 950 像素内是灵活的，但在超过这一宽度以后就不再继续伸展，因此我不会为大于该值的宽度指定媒体查询。注意这个 `max-width` 是布局的属性，并非像 `(max-width: 950px)` 中的媒体查询特性。

也可以使用与设备视觉区域宽度不完全相同的值作为断点。如果基于 `(min-width: 700px)` 的媒体查询是呈现内容的最佳方式，就可以使用该媒体查询。这样的话，也不一定要用像素作为单位。可以使用 em 创建媒体查询，如 `(min-width: 20em)`。

移动编码和测试工具

在移动电话和平板电脑上测试页面是一件很有挑战的事，因为你可能很难真正在各种设备上进行操作。在编码和开始最初的测试时，有一些值得使用的技巧和工具，尽管使用它们与在真实的设备上测试并不完全一样。

- ❑ 在编写样式的时候，调整桌面浏览器窗口的大小，使窗口大约等于各种移动电话和平板电脑的视觉区域大小。这肯定是一种相当粗略的方法，但它确实可以帮你测试样式，从而减少在设备上测试以后进行调整的工作量。此外，来回拖动浏览器窗口也可以测试页面布局为适应不同桌面浏览器进行调整的情况。

- 在开发的起步阶段，使用 ProtoFluid（www.protofluid.com）。这是一款免费的、基于浏览器的工具，提供符合一些流行设备尺寸的视图。需要强调的是，这款软件不能用于正式测试，因为它无法像手机或 iPad 那样运行，但它在开发的起步阶段还是很有帮助的。（一个小的警告是，该软件并不是一种非常直观的工具。你可能需要捣鼓一会儿才能掌握一些窍门。）要在 ProtoFluid 中看到页面，需要将页面存放到服务器上。服务器可以是由 Web 主机提供商提供的（参见第 21 章），也可以是运行在你自己的计算机上供开发用的服务器（在网上搜索“set up localhost server”）。
- 使用苹果免费提供的 iOS Simulator 测试页面在 iPhone 和 iPad 上的显示效果。现在我们已经有了一些基础，该软件是除了在真正的设备上测试以外最好的工具。你可能已经注意到了，我在一些屏幕截图上使用了这一工具。不过，该软件只能在 OS X 上运行，且在 Windows 上没有可替代的软件。iOS Simulator 是免费的 Xcode 的一部分，下载地址为 http://developer.apple.com/xcode/。
- 使用 Electric Mobile Simulator for Windows（www.electricplum.com/dlsim.html）。这可能是运行在 Windows 上最好的移动设备模拟软件了。当然，该软件并不是苹果的 iOS Simulator 的替代软件。
- 使用为其他设备和移动浏览器设计的仿真器和模拟器。Mobile Boilerplate 维护了一个针对 iOS、Android、Nokia Symbian 等设备的移动仿真器和模拟器列表，详情参见 https://github.com/h5bp/mobile-boilerplate/wiki/Mobile-Emulators-&-Simulators。

如果幸运的话，你身边的朋友可能有一些不同设备供你进行测试。四处问问吧！

4. 在 IE8 及以下版本中呈现媒体查询样式

对于先编写基准样式，再使用媒体查询对设计进行调整的做法，有一点需要注意，就是 Internet Explorer 8 及以下的版本不支持媒体查询。这意味着这些浏览器只会呈现媒体查询以外的样式，即基准样式。对大部分网站来说，使用 IE6、IE7 和 IE8 的用户合起来仍占有较大的比例，因此你可能希望这些访问者也能看到预期的设计。

Scott Jehl 着手修复了这一问题，他创建了一个轻量级的脚本，即 respond.js，该脚本可以让 `min-width` 和 `max-width` 媒体查询在 IE 的旧版本中生效。该脚本位于 https://github.com/scottjehl/Respond。激活 respond.min.js 链接可以查看代码，将代码复制到文本编辑器里，再保存为 respond.js 即可。

尽管这段脚本还可以让 `min-width` 和 `max-width` 媒体查询对其他旧浏览器有效，但其他旧浏览器并不重要，因此你可以使用条件注释仅让 IE8 及以下版本加载这段脚本，如图 12.3.11 所示。

```
...

        <footer id="footer"
role="contentinfo">
...
        </footer>
    </div>
</div>

<!--[if lte IE 8]>
    <script src="assets/js/respond.js">
    → </script>
<![endif]-->
</body>
</html>
```

图 12.3.11　将 script 元素放在条件注释里，只有 Internet Explorer 8 及以下版本会加载 respond.js。将 src 值里的 assets/js/ 部分替换为你的网站上指向 respond.js 的路径（如果与此不同的话）。设置完成后，IE8 及以下版本就能理解媒体查询并呈现相应的样式了

此外，如果你计划在项目中使用 Modernizr（www.modernizr.com），你可以通过 Modernizr 的配置工具将 respond.js 包含在内，从而不必单独下载并在 HTML 页面中调用这段脚本。关于 Modernizr 的更多信息，参见 14.3 节；关于脚本，参见第 19 章。

5. 构建适用于媒体查询的页面的步骤

(1) 创建内容和 HTML。

(2) 在 HTML 页面的 head 元素中，输入 `<meta name="viewport" content="width=device-width, initial-scale=1.0" />`。（参见 12.2 节的“理解视觉区域及使用视觉区域 meta 元素”。）

(3) 选择设计实现方法。推荐像示例演示的那样，先创建适用于所有设备的基准样式，再使用媒体查询逐渐进行完善。不过，如果你愿意，你也可以先实现桌面布局（参见第 11 章），再使用媒体查询为屏幕更小的设备定义样式。

(4) 为不同的视觉区域宽度调整适合内容的布局。根据 12.2 节中介绍的方法（参见图 12.2.3、图 12.2.5、图 12.2.7 和图 12.2.9），识别断点并创建相关联的媒体查询。在这个过程中，应尽可能地使用百分数表示 width、margin 和 padding，形成流式页面布局。

(5) 如果你想让 IE8 及以下版本呈现 min-width 和 max-width 媒体查询内的样式规则，可以获取 respond.js（参见“在 IE8 及以下版本中呈现媒体查询样式”）。

(6) 如果执行了第 (5) 步，应在 </body> 结束标记上面输入以下代码，从而让页面指向 respond.js（参见图 12.3.11）。

```
<!--[if lte IE 8]>
    <script src="path/respond.js">
    → </script>
<![endif]-->
```

其中 path 指的是你网站上 JavaScript 文件的位置，respond.js 是你保存的 JavaScript 文件的名称。script 元素周围的条件注释是可选的。

(7) 开始测试（参见“移动编码和测试工具”）。

(8) 根据需要，修改第 (4) 步中的 CSS 和媒体查询，并进行测试，直到页面在各种设备下都呈现出预期的效果。

提示　如图 12.3.3、图 12.3.5、图 12.3.7 和图 12.3.9 所示，所有示例页面的样式都位于一个样式表里。你也可以采用 12.2 节中指出的方法，链接到分离的样式表。通常，使用一个样式表的性能更优（只要它没有特别大），因为浏览器需要下载的文件越少，呈现页面所用的时间就越少。

提示　Eivind Uggedal 的 http://mediaqueri.es）网站汇集了大量现实的响应式站点（数量还在增加），值得借鉴。

提示 Mobile Boilerplate（www.html5boilerplate.com/mobile）是一个编写网页所用的起动模板，其中包含了很多移动开发的最佳实践。它背后的团队还针对现代移动设备（既包括智能手机，也包括平板电脑）提供了一组方便使用的规范（https://github.com/h5bp/mobile-boilerplate/wiki/Mobile-Matrices）。如该页面所述，这些信息对设计查询是非常有用的。

提示 如果你想定位高像素率设备（如 iPhone 4 和使用 Opera Mobile 11 浏览器的手机），可以使用以下媒体查询：

```
@media only screen and (-webkit-min-
→ device-pixel-ratio: 1.5), only
→ screen and (-o-min-device-pixel-
→ ratio: 3/2), only screen and
→ (min-device-pixel-ratio: 1.5) {
    /* 样式规则 */
}
```

这是 Mobile Boilerplate 推荐的方法。

提示 Andy Clarke 和 Keith Clark 创建的 320 and Up（http://stuffandnonsense.co.uk/projects/320andup/）是另一个用于起动的模板。它反映了这里演示的方法——从一套默认样式开始，逐渐使用媒体查询对布局进行放大。

提示 Luke Wroblewski 很好地总结了构建响应式站点有关事项的当前状态，见 http://www.lukew.com/ff/entry.asp?1436。他的文章还包含了一些扩展阅读材料的链接。

提示 Maximiliano Firtman 维护了一个现代移动设备对 HTML5 和 CSS3 支持情况的表格，见 http://mobilehtml5.org。（其中大量信息属于 HTML5 高级特性，本书未涉及这些内容。）

提示 如果 respond.js 不管用，可以查看 https://github.com/scottjehl/Respond 上的“Support & Caveats”（支持与注意事项）部分。如果还有问题，可以试着将图 12.3.11 中突出显示的代码放到页面的 </head> 结束标记之上，而非 </body> 结束标记之前。

提示 更多关于条件注释的信息，参见 www.quirksmode.org/css/condcom.html。

第 13 章

使用 Web 字体

本章内容

- 什么是 Web 字体
- 在哪里能找到 Web 字体
- 下载第一个 Web 字体
- 使用 `@font-face`
- 对 Web 字体添加样式及管理文件大小

在过去的几年中，我们看到了在万维网上使用字体的复兴。过去，我们通常只能在非常有限的字体集中进行选择，现在有了 Web 字体，我们在开发 Web 项目时有了大量字体可供选择。经历这种改变既让人着迷，又令人激动。

过去，选择字体相当简单。默认情况下，用户计算机上安装的字体非常有限，基本上就那几种。这就是大多数网站在 `body` 的样式中一成不变地指定 Georgia、Arial、Verdana 或 Trebuchet 的原因。这些字体在字号较小时看起来很不错，同时任何一台 Mac 或 Windows 计算机都安装了它们。

不夸张地说，现在，我们已经处于一个全新的世界。

13.1　什么是 Web 字体

CSS 规则 `@font-face` 为 Web 字体创造了可能，该样式规则允许 CSS 指向服务器上的一种字体供网页使用。

很多人认为 Web 字体是新生事物。实际上，Web 字体大约在 1998 年就产生了。Netscape Navigator 4 和 Internet Explorer 4 均采用了这种技术，但它们的实现都不支持标准的字体文件格式，因此很少有人用到它们。直到将近十年以后，浏览器才开始采纳这种使用更为常见的字体文件格式的标准，Web 字体的使用才变得常见起来。

1. Web 字体文件格式

Web 字体可以使用一系列文件类型。

- 内嵌 OpenType（Embedded OpenType，.eot）。在使用 `@font-face` 时，Internet Explorer 8 及之前的版本仅支持内嵌 OpenType。内嵌 OpenType 是 Microsoft 的一项专有格式，它使用数字版权管理技术防止在未经许可的情况下使用字体。
- TrueDoc。Netscape Navigator 4.0 最初采用了这种文件格式，但之后就不再采用也未获得支持。
- TrueType（.ttf）和 OpenType（.otf）。TrueType 和 OpenType 是桌面计算机广泛支持的标准字体文件类型，Mozilla Firefox（3.5 及之后的版本）、Opera（10 及之后的版本）、Safari（3.1 及之后的版本）、Mobile Safari（iOS 4.2 及之后的版本）、Google Chrome（4.0

及之后的版本）及 Internet Explorer（9 及之后的版本）均支持它们。

- 可缩放矢量图形（Scalable Vector Graphics，.svg）。这种格式用于其他格式均不被支持的某些特殊环境，如 Mobile Safari 的早期版本。
- Web 开放字体格式（Web Open Font Format，.woff）。这种较新的标准是专为 Web 字体设计的。Web 开放字体格式的字体是经压缩的 TrueType 字体或 OpenType 字体。WOFF 格式还允许在文件上附加额外的元数据。字体设计人员或厂商可以利用这些元数据，在原字体信息的基础上，添加额外的许可证或其他信息。这些元数据不会以任何方式影响字体的表现，但经用户请求，这些元数据可以呈现出来。Mozilla Firefox（3.6 及之后的版本）、Opera（11.1 及之后的版本）、Safari（5.1 及之后的版本）、Google Chrome（6.0 及之后的版本）及 Internet Explorer（9 及之后的版本）均支持 Web 开放字体格式。考虑到 Web 开放字体格式得到的广泛支持，这种格式可能会被选为行业标准。

2. Web 字体的浏览器支持情况

现代浏览器对 Web 字体的支持情况普遍很好。由于早期的浏览器仅支持特定的字体格式，因此 Web 开发人员需要付出一些额外精力应付这些早期浏览器，但回报是在所有的现代桌面浏览器甚至大多数智能手机浏览器上实现丰富的排版。

3. 法律问题

从技术层面上说，字体都是小的软件。我认识以字体设计和制作为生的人，这是一种艰苦、细致的创造性活动，没有强大的心智是做不了这份工作的。出于这一原因，即便 `@font-face` 功能从一开始就存在，这项工作还能激怒一些人也就不难理解了。毕竟，如果浏览器能链接并下载某种字体，意味着任何人都能下载并在自己的计算机上安装这种字体，无论他们是否购买了这种字体。这就是为什么 Web 设计人员和开发人员必须确保网站上用到的任何字体都具有在万维网上使用的许可。大多数厂商和字体服务提供商通过将许可作为字体购买的一部分或菜单选项提供这种许可。如果不进行购买，你可以将字体限定在免费字体的范围内，如可在 Font Squirrel（www.fontsquirrel.com）或 The League of Moveable Type（www.theleagueofmoveabletype.com）下载的字体。不管选用哪种方式，都要确保对项目中使用的 Web 字体具有稳定的权利。为此，你可以查看购买的字体的许可证。由于这是近期的一个热点话题，你要购买字体的厂商的网站通常会提到这些信息。如果存在疑惑，可以跟厂商联系，了解哪些权利是允许的。

如果你购买了一个字体，并确切地知道可以将其用做 Web 字体，就可以使用 Font Squirrel 提供的免费 `@font-face` 生成器（www.fontsquirrel.com/fontface/generator）。该工具可以将字体转化为在万维网上使用的所有 Web 字体文件类型。

13.2　在哪里能找到 Web 字体

在网站上使用 Web 字体有两种方式：自托管和 Web 字体服务。这两种方式都是完全有效的选项，不过二者差异很大，各有利弊。当你权衡这些利弊的时候，你会发现并非所有的 Web 字体都是随处可见的。你可能会发现，即便你想采用自托管的方式，你需要的

字体也只能来自 Web 字体服务。这时，你可能需要寻找一种接近的替代方案，或者重新考虑你的方法。在作出决定之前，需要花费一些精力权衡所有的选项并灵活处理。

1. 自托管

自托管 Web 字体是一种更为传统的方式，也是本章将逐步进行讲解的方式。字体来源于你自己的服务器，就像其他的资源（如图像、CSS 文件）一样。如果需要与字体相关的花费，通常也是一次性的购买支出，是否将字体文件上传到网站上并将其包含到代码中取决于你自己。

寻找可以自托管的字体是比较容易的，因为它们的数量很多。它们的质量和价格差异很大（有的甚至是免费的）。其中一些流行的自托管字体来源如下。

- Font Squirrel（www.fontsquirrel.com）
- MyFonts（http://myfonts.com）
- The League of Moveable Type（www.theleagueofmoveabletype.com）
- FontShop（www.fontshop.com）

2. Web 字体服务

Web 字体服务通常提供订购 Web 字体的方法。这种方式是指按月或按年支付使用字体的版权费用，而不是彻底买断字体。这些服务托管字体，为用户提供一小段代码放在其网页中。这段代码可能是 JavaScript，也可能是 CSS，这取决于服务自身。它包含了从远程服务器获取字体文件并将其显示在网站上这一过程所需的全部代码。很多人主张使用这种方式，因为这种方式通常比单独购买字体更便宜，而且它让用户有机会尝试很多种不同的字体。

一些流行的 Web 字体服务包括：

- Typekit（https://typekit.com）
- Fontdeck（http://fontdeck.com）
- Fonts.com（www.fonts.com）
- WebINK（www.webink.com）
- Google Web Fonts（www.google.com/webfonts）

按理说，Web 字体服务可以提供比自托管更多的特性。所有的东西都存放在服务器上，包括字体文件。如果出现了更好的字体文件或代码，服务便可以很容易地将它们提供给用户。

此外，在这类服务中，很多都使用 JavaScript 嵌入获取 Web 字体的代码。这样做有一些好处，也有一些不好的地方。在这一过程中，JavaScript 做了不少工作，包括探测加载页面的浏览器，为字体本身的加载提供额外的控制。这种控制可以产生更好的体验，因为它让 Web 字体服务可以自定义字体格式和提供字体的代码。例如，Typekit 近期宣称，它们将在一些服务中使用基于 PostScript 的轮廓（仅针对 Windows 浏览器）显示的字体，从而让笔画更为平滑（http://blog.typekit.com/2011/09/15/improvedwindows-rendering-for-more-typekit-fonts/）。如果没有 JavaScript，这类特性就失效了。

这种奢华的代价当然是你必须百分之百地依赖 JavaScript。如果用户在其浏览器中禁用了 JavaScript，他们就无法看到你的 Web 字体。此外，JavaScript 会影响页面的性能。在 Web 字体呈现在页面上之前，用户必须等待 JavaScript 加载完成。这是你决定如何在网站上加入 Web 字体时需要记住的事情。

3. Web 字体的质量与显示

不幸的是，并非所有的 Web 字体都是一样的。对于在不同浏览器中 Web 字体显示的效果，它们具有很明显的差异。一些字体在

Internet Explorer 较早的版本中显示效果并不太好，这一点尤为明显。

当你选择字体时，要尽量检查你即将选择的字体在各种浏览器中显示的效果（如图 13.2.1 所示）。现在，由于很多 Web 字体公司提供 Web 字体的实时示例，有的公司提供字体在不同浏览器和平台中的截图，因此上述工作就变得容易多了。



图 13.2.1　这张合成的截图显示的是相同的代码在 Internet Explorer 6（上）和 Chrome 15（下）中显示的效果。可以注意到，在 Internet Explorer 中，字母的笔画更细，且不太平滑

提 示　如果你坚持自己测试，可以尝试 Web Font Specimen（http://webfontspecimen.com）提供的工具。这个工具可以让你查看 Web 字体在不同环境和不同字号下显示的效果。

13.3　下载第一个 Web 字体

下载免费的 Web 字体很快也很方便。我们将使用 Font Squirrel，它提供了一个 demo.html 文件让你查看这些字体在实践中的效果。

下一节，我们将研究 `@font-face` 的语法以及如何将 Web 字体集成到页面中。

1. 从 Font Squirrel 下载 Web 字体

(1) 进入 Font Squirrel 的 `@font-face` 包板块（www.fontsquirrel.com/fontface），选择你想使用的字体。在这里，我选择了 League Gothic。

(2) 点击 Get Kit（获取包）链接，如图 13.3.1 所示，就会立即开始下载。下载的文件是一个 ZIP 压缩包。

(3) 下载完成后，打开压缩包，就会看到一个包含 Web 字体、CSS 文件和 demo.html 文件的文件夹，如图 13.3.2 所示。



图 13.3.1　如果你想在下载之前进一步查看字体，可以点击 View Font（查看字体）了解关于字体的更多信息，以及下载选项。如果点击 View @ff Demo（查看 @ff[1] 示例），则可以看到使用这种字体作为 Web 字体时的完整范例，这是一种快速测试字体在不同浏览器中显示效果的方法

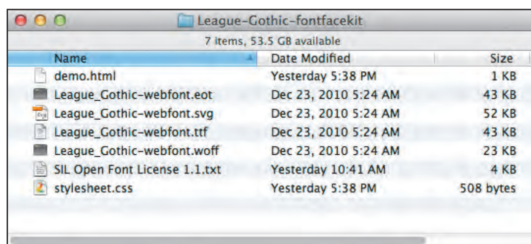


图 13.3.2　League Gothic 的 ZIP 压缩包展开后的内容。如你所见，有一个 demo.html 文件、四个 Web 字体、一个许可证和一个样式表

2. 在 demo.html 文件中查看所选字体

在浏览器中打开位于所下载字体文件夹里的 demo.html 文件，如图 13.3.3 所示（参见 2.7 节）。

[1] @ff 指的是 `@font-face`。——译者注

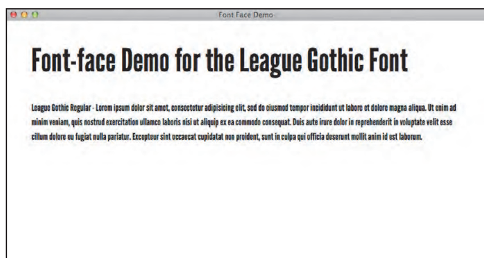


图 13.3.3　看！ Web 字体尽显魅力

这个演示文件显示，Web 字体确实起作用了。这是非常令人激动的！在你宣布胜利并准备收工之前，我们还是要在下一节讲讲其中的工作原理。

提示　你开始下一节项目之前，想获得字体选择的灵感吗？ Typekit 团队写了一篇很好的博客文章，这篇文章介绍了大量关于 Web 字体及一般性排版的有用信息。初学者可以尝试阅读“Sites we like”（我们喜爱的站点）系列，参见 http://blog.typekit.com/category/sites-welike/。

提示　需要在 Photoshop 中使用这些字体吗？你可以将 Web 字体包中的 TrueType（.ttf）字体装到你自己的计算机上。安装完成后，你就可以像使用其他字体一样使用该字体了。

13.4　使用 @font-face

你已经下载了 Web 字体包，并在浏览器中测试了 demo.html 文件。现在，不妨看看其中的原理。首先，让我们看看 stylesheet.css 中的代码（参见图 13.4.1）。

如你所见，样式表相当简单，只有一条样式规则。不可否认的是，这条规则相当重要！

`@font-face` 语法与传统的 CSS 相比有些不同。其中的一点就是，它并未像你从第 8 章起接触到的传统方式那样，一个选择器后面跟着一些属性 / 值对。这条样式规则以看起来有些奇怪的 `@font-face` 声明开头。

```
@font-face {
font-family: 'LeagueGothicRegular';
src: url('League_Gothic-webfont.eot');
src: url('League_Gothic-webfont.eot?#iefix')
format('embedded-opentype'),
        url('League_Gothic-webfont.woff')
        ➝ format('woff'),
        url('League_Gothic-webfont.ttf')
        ➝ format('truetype'),
        url('League_Gothic-webfont.
        ➝ svg#LeagueGothicRegular')
        ➝ format('svg');
font-weight: normal;
font-style: normal;
}
```

图 13.4.1　这是 Font Squirrel 在字体包中提供的 `@font-face` 样式规则。可以注意到，它使用了单引号，而不是本书其他 CSS 示例中使用的双引号。在 CSS 中，单引号和双引号的效果是一样的，因而你可以根据自己的偏好进行选择

为了搞清楚 `@font-face` 的工作原理，可以将 `@font-face` 样式规则理解成建立一种为其他 CSS 所用的工具。这条规则并不影响任何特定元素的样式，但它为在 CSS 中使用 Web 字体打下了基础。

这条样式规则的第一行是对字体系列的指定：

```
font-family: 'LeagueGothicRegular';
```

这条声明指出了这种 Web 字体的名称。在本例中，我们使用 `LeagueGothicRegular`，而实际上它可以是你选择的任何字体的名称。你可以选择 `Banana` 或 `The Best Font Ever`，这取决于你自己。

这条样式规则的下面几行告诉浏览器字体文件的位置。其中包括为所有支持 Web 字体的不同浏览器准备的字体文件格式。这些语法看起来有些吓人，但我们的目的并不是

要完全地理解它们。如果你想进一步深入研究，了解这样显示的原理，我向你推荐 Ethan Dunham 发表在 Fontspring 上的博文（www.fontspring.com/blog/further-hardening-of-the-bulletproof-syntax）。作者在这篇文章中解释了他对 `@font-face` 语法的最新思考。

1. 将 Web 字体包含到网页里

我们讲过了 `@font-face` 的语法，但还从来没有真正地将 Web 字体放到网页里去。让我们看看 demo.html 的代码，并研究一下位于页面顶部的 CSS 代码（如图 13.4.2 所示）。

```
h1.fontface {
    font: 60px/68px 'LeagueGothicRegular',
    → Arial, sans-serif;
letter-spacing: 0;
}

p.style1 {
    font: 18px/27px 'LeagueGothicRegular',
    → Arial, sans-serif;
}
```

图 13.4.2　这是 demo.html 文件顶端的 CSS。Font Squirrel 将它放在这个位置，是出于演示的目的。在实践中，最好将所有的 CSS 放在一个外部样式表中

这些规则为 HTML 添加了 League Gothic Web 字体的样式。在第一条样式规则中，`60px/68px` 指定了字体大小和行高。在示例中我们以像素为单位指定大小，但你也可以使用其他的单位。在字体大小和行高之后指定了 `'LeagueGothicRegular', Arial, sans-serif`。如 10.2 节所述，可以像这样指定多个字体系列（以逗号分隔），即所谓的字体栈。如果浏览器不支持字体栈中的第一个字体，便会尝试下一个字体。由于我们使用的是 Web 字体，因此浏览器很可能如期显示该 Web 字体，但使用字体栈仍然是一种很好的实践方式。毕竟，并非所有的浏览器都支持 Web 字体。

在 CSS 中，`font` 属性引用了 `LeageGothicRegular`，这也是 `@font-face` 样式规则中指定的 `font-family` 名称。更重要的是，它是像指定其他字体一样指定这一字体的。在浏览器看来，League Gothic 就像是在安装在网站访问者计算机上的字体。

既然你已测试成功，或许你想再尝试一些其他的 Web 字体？

我们的下一项任务就是为主标题使用 League Gothic 字体，为页面的其余文字使用 Crimson 字体。

2. 使用多种 Web 字体的方法

回到 Font Squirrel，下载 Crimson Web 字体包，其中包含了六种不同的字体，如图 13.4.3 所示。



图 13.4.3　如你之前做过的，点击 Get Kit 下载该 Web 字体包的 ZIP 压缩包。下载完成后，将压缩包解压

3. 在同一个项目中使用多种 Web 字体

(1) 在字体包文件夹中有很多字体。我们只需要其中的少数几个。选中文件名为 Crimson-Roman-webfont 的四个 Web 字体文件，将它们复制到我们之前操作过的 League Gothic 文件夹。结果应如图 13.4.4 所示。

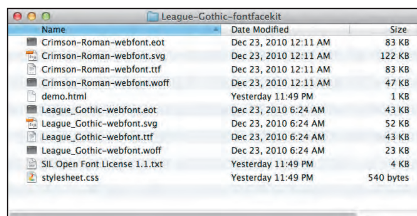


图 13.4.4　现在，我们在同一文件夹中有了两种 Web 字体。好样的！（关于文件的组织，参见最后一条提示。）

(2) 在 stylesheet.css 中输入以下样式规则：

```
@font-face {
    font-family: "CrimsonRoman";
    src: url("Crimson-Roman-
    → webfont.eot");
    src: url("Crimson-Roman-
    → webfont.eot?#iefix")
    → format("embedded-opentype"),
        url("Crimson-Roman-webfont.
        → woff") format("woff"),
        url("Crimson-Roman-webfont.
        → ttf") format("truetype"),
        url("Crimson-Roman-webfont.
        → svg#CrimsonRoman")
        → format("svg");
}
```

我敢肯定，这些代码看起来非常熟悉。现在，我们的样式表里既有 League Gothic 字体的样式规则，也有 Crimson Roman 的样式规则。下一步是添加选择器，将 Crimson 字体用到页面中去。

(3) 在你刚刚输入的 `@font-face` 样式规则后面，从新的一行输入以下内容：

```
body {
    font-family: "CrimsonRoman",
    → Georgia, serif;
}
```

(4) 在新的一行，输入以下样式规则，对 `h1` 添加样式：

```
h1 {
    font-family: "LeagueGothic
    → Regular", Arial, sans-serif;
    font-size: 4em;
    font-weight: normal;
}
```

接下来，我们将创建一个新的 HTML 文档。

(5) 在同一个文件夹，创建一个新的 HTML 文件，并命名为 demo2.html。

(6) 在 demo2.html 页面中输入以下代码（注意这里是如何指向我们刚才编辑的 stylesheet.css 的）：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Our Awesome Web Font
    → Examples</title>
    <link rel="stylesheet"
    → href="stylesheet.css" />
</head>
<body>
<article>
    <h1>Headlines Are Very
    → Important</h1>
    <p>There is more to Philadel
    → phia than cheesesteaks,
    → Rocky and the Liberty Bell.
    → Did you know that Phila
    → delphia used to be the
    → capital of the United States?
    → You will also find that
    → Philadelphia has our nation's
    → first Post Office, Hospital,
    → and free library. That Ben
    → jamin Franklin was one busy
    → fellow!</p>
</article>
</body>
</html>
```

(7) 在 Web 浏览器中打开 demo2.html，如图 13.4.5 所示（参见 2.7 节）。现在，你的字体看起来应该很不错。

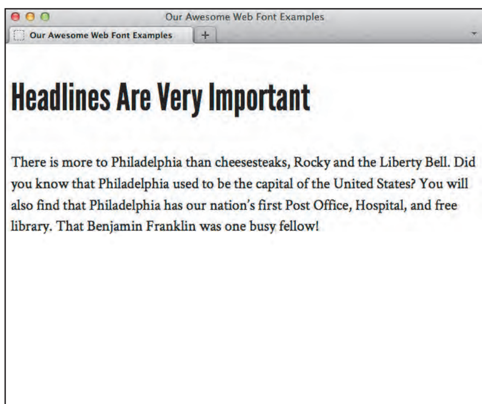


图 13.4.5　标题使用 League Gothic 字体，页面的其余文字使用 Crimson 字体

提示　我们使用了来自 Font Squirrel 的字体包里的字体系列名称，如 `LeagueGothic-Regular`。这是很好的描述性名称，但另一种学派认为应该使用更为语义化的名称，如 `font-family: "Headline";`。选择这种方案的一个好处是，如果你想改主意，选择另一种 Web 字体（而不是 League Gothic），你只需要替换 @font-face 样式规则，并将字体系列的名称指定为 `Headline` 即可。

提示　尽管上述步骤和示例中假定字体文件、样式表和 HTML 页面均位于同一个目录，但在实践中最好使用文件夹对它们进行组织（参见 2.6 节）。一定要根据需要修改 HTML 中指向样式表的路径，以及 CSS 中指向 Web 字体的路径（参见 1.7 节）。

13.5　对 Web 字体添加样式及管理文件大小

Web 字体比常规字体要复杂一些，因此在操作 Web 字体的时候，有一些要注意的地方。

使用 Web 字体（尤其是使用多个 Web 字体）的潜在风险之一是，它们可能会增加页面的“重量”。我在这里谈论的不是熏肉和甜甜圈，我谈论的是千字节和兆字节。

对于所有这些字体来说，在它们显示到页面上之前，都需要下载到用户的计算机上。如果一个页面上有五六个字体，它们就会减慢网站加载的速度，尤其是对移动用户来说。我的建议是审慎地选择 Web 字体。如果你发现自己使用了七个 Web 字体，你就应该想办法对这些字体进行整合了。

节省页面重量的一种方法是构造子集（subsetting）。构造子集是通过仅包含确定使用的字符削减实际字体大小的方法。例如，如果对标题使用 League Gothic 字体，而网站的设计要求标题始终使用大写字母，没有使用小写字母的需要。通过构造子集，可以将小写字母从字体中移除，这样字体文件大小就会减小一些。

此外，对于很多字体，你还可以选择针对特定语言的子集。在 Font Squirrel 浏览字体的时候，对特定的字体点击 View Font 而非 Get Kit，就可以在下载字体之前选择“`@font-face` 包”查看针对特定语言的选项。解释子集构造的具体细节已经超出了本书的范围，不过 Font Squirrel 提供了帮助你进行专家级子集构造的工具（www.fontsquirrel.com/fontface/generator）。

Web 字体用起来有些奇怪的另一种情景出现在当你想对它们添加一些基本样式的时候。需要记住的是，对于 Web 字体，每个字体只有一种粗细和一种风格。如果你想使用粗体或斜体，就需要为它们创建单独的样式规则，每条规则对应一个 Web 字体文件。

添加斜体和粗体的步骤

(1) 使用以下代码中突出显示的部分，对 demo2.html 文件的第一个段落进行更新。

```
<p>There is more to Philadelphia
→ than cheesesteaks, Rocky and
→ the Liberty Bell. <em>Did you
→ know</em> that Philadelphia
→ used to be the capital of the
→ United States? You will also
→ find that Philadelphia has
→ our nation's first Post Office,
→ Hospital, and free library.
→ That Benjamin Franklin was one
→ <b>busy</b> fellow!</p>
```

(2) 刷新 Web 浏览器。

段落中出现了粗体和斜体的样式。不过，事情并不像看起来的这样完美。实际上，Crimson Roman Web 字体并没有内置的粗体和斜体样式，浏览器是通过将常规文本的笔划变粗一些来模拟粗体的，通过将常规文本稍微倾斜一些来模拟斜体的。

这种假斜体的效果是很明显的，即便是一般的人也能看出来（参见图 13.5.1）。我们需要的是为特定字体专门设计的合理的粗体和斜体。好消息是，使用这样的粗体和斜体很简单!

This is fake italic. It is not real.

This is not fake italic. It is real.

图 13.5.1　哪一个是假斜体？注意小写字母 k、a 和 f

要对 Crimson 字体使用合理的粗体和斜体，必须先获得这些 Web 字体文件，再将它们复制到“在同一个项目中使用多种 Web 字体的步骤”中使用的文件夹。

(3) 找到 Crimson-Bold 和 Crimson-Italic Web 字体文件，将它们的全部格式（总共有八个文件）复制到演示项目文件夹（参见图 13.5.2）。

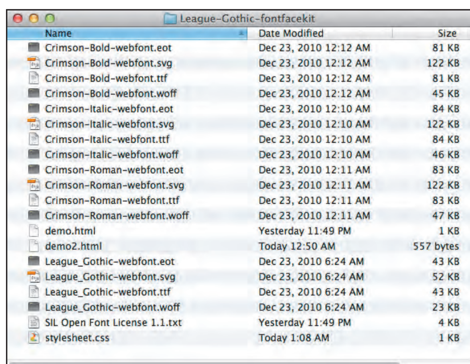


图 13.5.2　天哪，这里竟然有了这么多文件！新的 Crimson 字体的粗体和斜体文件应该位于 Roman 文件的旁边。（参见最后一条提示。）

接下来，跟以前一样，需要添加一些新的 `@font-face` 样式规则将粗体和斜体文件添加到网页中去。

(4) 在 stylesheet.css 中输入以下内容：

```
@font-face {
  font-family: "CrimsonBold";
  src: url("Crimson-Bold-webfont.
  → eot");
  src: url("Crimson-Bold-webfont
  → .eot?#iefix") format(
  → "embedded-opentype"),
     url("Crimson-Bold-webfont.
     → woff") format("woff"),
     url("Crimson-Bold-webfont.
     → ttf") format("truetype"),
     url("Crimson-Bold-
     → webfont.svg#CrimsonBold")
     → format("svg");
}

@font-face {
  font-family: "CrimsonItalic";
  src: url("Crimson-Italic-
  → webfont.eot");
  src: url("Crimson-Italic-
  → webfont.eot?#iefix")
  → format("embedded-opentype"),
     url("Crimson-Italic-webfont.
     → woff") format("woff"),
```

```
        url("Crimson-Italic-webfont.
        ➝ ttf") format("truetype"),
        url("Crimson-Italic-webfont.
        ➝ svg#CrimsonItalic")
        ➝ format("svg");
}
```

(5) 将以下样式规则添加到 stylesheet.css 中：

```
b {
    font-family:"CrimsonBold",
    ➝ Georgia, serif;
    font-weight: normal;
}

em {
    font-family:"CrimsonItalic",
    ➝ Georgia, serif;
    font-style: normal;
}
```

第一条样式规则为 b 元素设置了样式，将字体系列设成了 Crimson Bold，并将 font-weight 设成了 normal。如果你忘了将 font-weight 设为 normal，保持粗细设置不变，那么浏览器就会让粗体更粗，这就更糟糕了！对 em 元素的处理也是类似的，即修改字体系列，并将 font-style 设为 normal。就应该这样做。

(6) 在 Web 浏览器中查看 demo2.html，可以看到合适的斜体和粗体（参见图 13.5.3）。

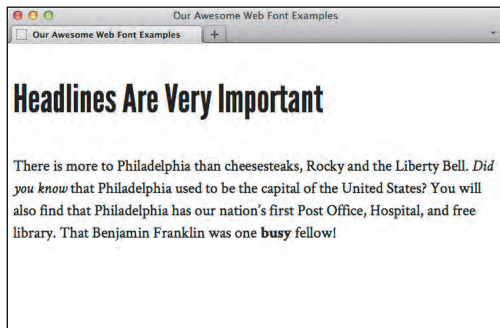


图 13.5.3　这里有多个 Web 字体，它们都有合理的样式

提 示　可以对任何包含文本的元素使用 Web 字体的粗体和斜体样式，并非只有示例和步骤演示中的那些元素。同往常一样，选择最能准确描述内容的 HTML 元素，再根据需要对其添加样式。

提 示　记住，每种风格和粗细都需要添加新的字体文件，这会增加浏览器需要下载的文件的大小，从而对性能产生影响。出于这一原因，很多设计人员仅对标题使用 Web 字体。

提 示　有一种更为简洁的书写 @font-face 代码的方法，它让你拥有额外的粗细和风格变化，而不必为 b、em 或其他任何需要添加粗体或斜体的元素编写额外的样式规则。不过，这样做也有一些额外的风险，因为 Internet Explorer 并不支持这种方法。要了解这种方法，可以阅读 Roger Johansson 的博客文章（www.456bereastreet.com/archive/201012/font-face_tip_define_font-weight_and_font-style_to_keep_your_css_simple/）。

提 示　如果使用 Typekit 或其他的使用 JavaScript 的 Web 字体服务，需要查看使用它们添加字体样式的方法，因为它们可能拥有自身的编写选择器的方式。

提 示　尽管步骤和示例中均假定字体文件、样式表和 HTML 页面位于同一个目录（如图 13.5.2 所示），但在实践中最好使用文件夹对它们进行组织（参见 2.6 节）。一定要根据需要修改 HTML 中指向样式表的路径，以及 CSS 中指向 Web 字体的路径（参见 1.7 节）。

第 14 章

使用 CSS3 进行增强

本章内容

- ❑ 理解厂商前缀
- ❑ 浏览器兼容性速览
- ❑ 使用 polyfill 实现渐进增强
- ❑ 为元素创建圆角
- ❑ 为文本添加阴影
- ❑ 为其他元素添加阴影
- ❑ 应用多重背景
- ❑ 使用渐变背景
- ❑ 为元素设置不透明度

网站制作者多年来面临的挑战之一就是，使用 CSS 建立丰富布局的选择是很有限的。在大多数情况下，要建立丰富的布局，就需要使用额外的 HTML、CSS 及大量图像。这样做的结果就是，页面变得更为复杂，可访问性降低，浏览器需要花更长的时间下载和显示，同时，页面变得更为脆弱，更难维护。

近年来，浏览器快速吸纳了很多新的 CSS3 属性，让上述情况有了改观。如今，仅使用 CSS 创建圆角、渐变和阴影以及调整透明度等已经变成现实，这也减少了网页需要使用的标记和图像的数量。或许更为重要的是，这让页面在处理能力较弱的设备（如智能手机）以及台式机、笔记本电脑上下载和显示的速度变得更快了。

随着 CSS 的持续发展，不同浏览器对新的CSS属性的支持程度不一仍然是一项挑战。

在本章中，我们将看到如何使用一些流行且实用的 CSS3 属性创建圆角、阴影和渐变，对一个元素使用多重背景，以及调整透明度。我们还将看到浏览器厂商和 Web 专业人员如何使用渐进增强的哲学，通过厂商前缀和基于 JavaScript 的 polyfill 缩小浏览器之间的差异。

本章的代码示例可以在本书的配套网站上找到（www.bruceontheloose.com/htmlcss/examples/）。该网站上还包括结合本章讨论的 CSS3 效果的其他一些示例。

14.1　理解厂商前缀

尽管很多属于 CSS3 范畴的特性都还未进入 W3C 的候选推荐标准阶段（意味着相应的规范已经完成），它们的大部分都已经被 Firefox、Internet Explorer、Chrome、Safari 和 Opera 的新版本实现了。

尚未完成的（以及偶尔存在竞争的）CSS 实现是有可能发生变化的，为了应对这种情况，浏览器在实现一些 CSS 特性时使用了所谓的厂商前缀。这样，每个浏览器都可以引入自己的 CSS 属性支持方式，这种方式不会与最终的规范发生冲突，也不会与其他浏览器发生冲突。此外，厂商前缀还能确保将来规范成熟或定稿时，现有的使用实验性实现的网站不会崩溃。

每个主流浏览器都有其自身的前缀：-webkit-（Webkit/Safari/Chrome）、-moz-（Firefox）、-ms-（Internet Explorer）、-o-（Opera）以及 -khtml-（Konqueror）。你可能想得到，它们将放置在 CSS 属性名的前面。不过要记住的是，不必每次都使用全部的前缀。在大多数情况下，只需要 -moz- 和 -webkit- 前缀，就像你会在本章示例中看到的那样。为了更好地面向未来，推荐在最后写上一条无前缀版本的声明，如图 14.1.1 所示。

```
div {
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    border-radius: 10px;
}
```

图 14.1.1　使用 border-radius 属性的示例。该属性需要使用厂商前缀（如前两条声明所示）支持旧版本的 Firefox 和基于 Webkit 的浏览器（如 Chrome 和 Safari）。这些浏览器的新版本已不再使用有前缀的属性，开始使用无前缀的属性（即如 border-radius: 10px;）。照例，样式规则中的最后一条声明优先级最高，这也是将无前缀版本放在最后的原因

在本章中你将看到，这样做意味着用较少的 HTML，而用更多的 CSS 实现所需的效果。

虽然厂商前缀通常会在 CSS 中造成大量的重复，但这只是前进路上的一点点代价，同时也是 Web 专业人士已经普遍接受的代价，因为他们已经找到了将在代码中添加前缀的繁琐工作自动化的方法（参见第后一条提示），如图 14.1.2 所示。

提 示　并非所有的 CSS3 属性都需要使用为浏览器准备的前缀，如 text-shadow 和 opacity（分别参见 14.5 节和 14.9 节）。此外，并非所有的浏览器都需要在属性中使用特定的前缀。



图 14.1.2　CSS3 Generator 既是一种实用的学习工具，同时可以用于减少书写有前缀的和无前缀的 CSS 属性这种重复的工作

提 示　如果属性的语法在定稿之前发生了改变，你可以在 CSS 中写上多个版本。可以在本章对 border-radius 和 gradient 属性的演示中找到有关的示例（分别参见 14.4 节和 14.8 节）。每种浏览器只会解释它理解的语法，而忽略其他的语法。

提 示　尽管对需要包含的有前缀属性的顺序没有要求，但应该总是在最后包含该属性的无前缀版本，以保证未来仍然可用（参见图 14.1.1）。这样，即便浏览器未来支持无前缀的属性，也不会产生中断的情况。

提 示　CSS3 Generator（www.css3generator.com）等服务可以简化创建这类属性的工作，减少输入，节约时间。其他用于生成代码的工具的列表参见 www.bruceontheloose.com/tools/。

14.2　浏览器兼容性速览

由于浏览器发展的脚步在近几年明显地加快了，因此理解这些新的 CSS 属性在

什么时候会得到预期的效果就很重要了。这里给出了浏览器对本章将要介绍到的属性开始提供基本支持的时间，如图 14.2.1 所示。

border-radius	1.0	9.0	1.0	3.0	10.3
box-shadow	3.5	9.0	1.0	3.0	10.5
text-shadow	3.0	10.0	1.0	1.1	10.0
多重背景	3.6	9.0	1.0	1.3	10.0
渐变	3.6	10.0	2.0	4.0	11.1
不透明度	1.0	9.0	1.0	2.0	10.0

图 14.2.1　这个表格显示了浏览器第一次支持本章将要讨论的各个 CSS 属性的时间。要获得更详细的分类，可以在 www.caniuse.com 或 www.findmebyip.com/litmus/ 查找各个属性

14.3　使用 polyfill 实现渐进增强

渐进增强是如今被普遍接受的一种建立网站的方式，它强调创建所有用户都能访问（无论使用什么样的 Web 浏览器）的基本层面的内容和功能，同时为更强大的浏览器提供增强的体验。简单地说，渐进增强意味着网站在不同 Web 浏览器中的外观和行为不一样是完全可以接受的，只要内容是可访问的。

Dribbble 网站（http://dribbble.com）就是渐进增强的一个实例，如图 14.3.1 所示。通过渐进增强，该网站使用 CSS3 为现代浏览器提供更丰富的体验。在旧的浏览器（如 Internet Explorer 8）中，该网站呈现出稍微不同的视觉体验，但功能并未削减，如图 14.3.2 所示。

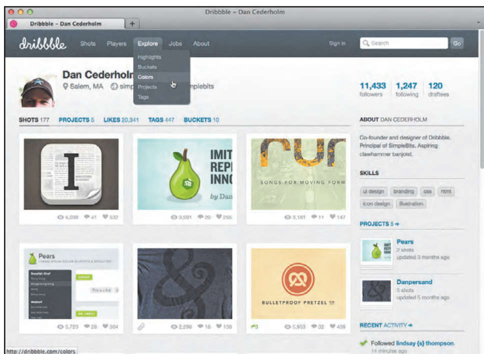


图 14.3.1　Dribbble 网站使用了一些 CSS3 属性，如 border-radius 和 CSS3 渐变背景，为使用现代浏览器的用户提供更为丰富的体验，同时，它也记得为功能较弱的浏览器着想

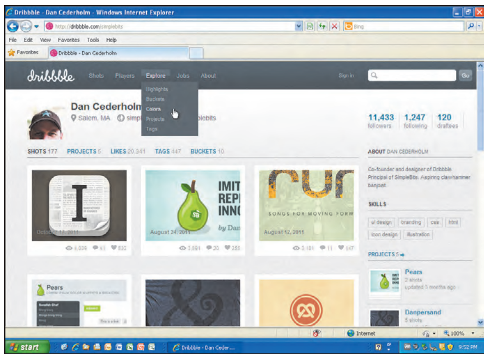


图 14.3.2　在不支持 border-radius 的旧的浏览器（如 Internet Explorer 8）中查看该网站，体验就会发生改变。圆角（如那些药丸状的导航按钮的圆角）变成直角了，但功能并未损失，一切照常工作。这是渐进增强在实践中的一个方面

如果你想弥合较弱的浏览器与较强的浏览器之间的功能差异，可以使用 **polyfill**（通常又称垫片，shim）。

polyfill 通常使用 JavaScript 实现，它可以为较弱的浏览器提供一定程度的对 HTML5 和 CSS3 的 API 和属性的支持，同时，当浏览器本身就具有相应的能力时，会不动声色地退而使用官方的支持。需要注意的是，这样做通常会对性能产生一定的负面影响，

因为较弱的 Web 浏览器（尤其是旧版本的 Internet Explorer）运行 JavaScript 的速度要慢得多。

Modernizr 的网站（www.modernizr.com）上可以找到更多关于渐进增强的信息，包括负责填补旧的浏览器与新的 Web 技术之间差异的各类 polyfill。Modernizr 是一个 JavaScript 库（参见图 14.3.3），由 Faruk Ateş 于 2009 年创建。目前团队成员还包括 Paul Irish、Alex Sexton 和 Ryan Seddon。更多关于 Modernizr 的信息参见提示。

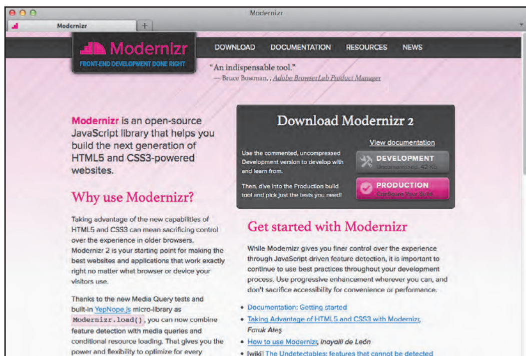


图 14.3.3　Modernizr 是一个 JavaScript 库，它允许你探测浏览器是否支持创建优化的网站体验所需的特定的 HTML5、CSS3 及其他的特性

提 示　如今，浏览器要么鼓励用户定期下载新的版本（Firefox、Safari 和 Internet Explorer 均采用这种方式），要么悄无声息地在后台下载更新（Chrome 就采用这种方式）。

提 示　当新的或实验性的 CSS 可用时，基于 JavaScript 的工具（如 Modernizr）可以提供暗示，从而允许你使用 CSS 和 JavaScript 对页面进行渐进增强，为更强的浏览器提供更为丰富的体验，同时不至于扔下其他的浏览器不管。

提 示　GitHub 上有一份由社区维护的关于实用的 JavaScript polyfill 的列表（https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills），如图 14.3.4 所示。这份列表也是 Modernizr 项目的一部分。其中，CSS3 Styles 部分里的项目尤其有趣，如 Jason Johnston 的 PIE（www.css3pie.com），它可以为 Internet Explorer 6 ~ 9 针对很多本章将要介绍的 CSS3 效果提供支持（其中，IE9 仅在需要显示线性渐变时用得上 PIE，对于其他的效果，IE9 具有原生的支持）。注意，使用 PIE 可能影响网站在这些旧的浏览器中运行的性能。

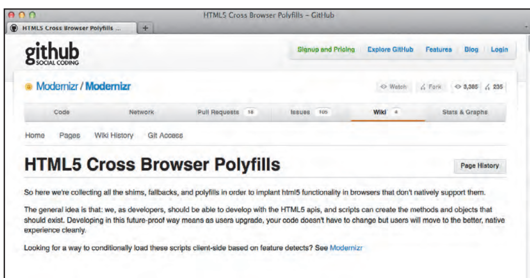


图 14.3.4　不断增长的 JavaScript polyfill 列表。JavaScript polyfill 可以为并不原生支持 HTML5 和 CSS3 特性的旧的浏览器提供这方面的支持

14.4　为元素创建圆角

使用 CSS3，可以在不引入额外的标记或图像的情况下，为大多数元素（包括表单元素、图像，甚至包括段落文本）创建圆角，如图 14.4.1 和图 14.4.2 所示。同 `margin` 和 `padding` 属性一样，`border-radius` 属性也有长短两种形式的语法。图 14.4.5 中有一些演示 `border-radius` 属性不同用法的基本示例。

```
...
<body>
<div class="all-corners"></div>
<div class="one-corner"></div>
<div class="elliptical-corners"></div>
<div class="circle"></div>
</body>
</html>
```

图 14.4.1 这个文档包含带 class 属性的示例 div。每个 div 均用于演示 border-radius 的不同用法和不同语法，包括创建相同的四个圆角，使用长形式语法创建单独的圆角，创建椭圆形圆角以及创建圆形等图形

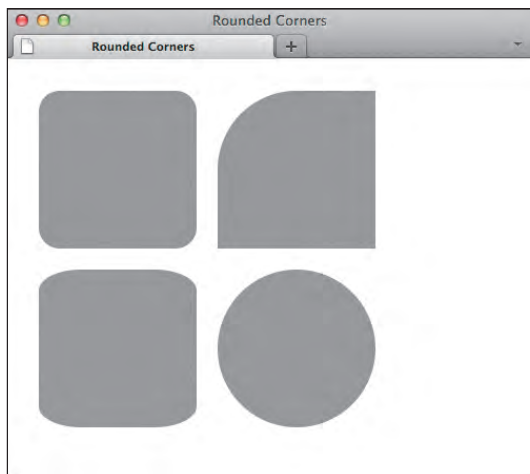


图 14.4.2 支持 border-radius 属性（包括有前缀的和无前缀的）的浏览器会以如上图所示的方式将示例呈现出来。注意，不同的实现之间可能存在视觉上的差异，尤其是在旧版本的 Safari 和 Firefox 中，差异更为明显

1. 为元素创建四个相同的圆角

(1) 输入 -moz-border-radius: *r*，这里的 *r* 是圆角的半径大小，表示为长度（带单位），如图 14.4.3 所示。

(2) 输入 -webkit-border-radius: *r*，这里的 *r* 是圆角的半径大小，使用与第 (1) 步中相同的值。

(3) 输入 border-radius: *r*，这里的 *r* 是圆角的半径大小，使用与第 (1) 步中相同的值。这是该属性的标准短形式语法。

```
div {
    background: #999;
    float: left;
    height: 150px;
    margin: 10px;
    width: 150px;
}

.all-corners {
    border-radius: 20px;
}

.one-corner {
    -moz-border-radius-topleft: 75px;
    -webkit-border-top-left-radius: 75px;
    border-top-left-radius: 75px;
}

.elliptical-corners {
    -moz-border-radius: 40px / 20px;
    -webkit-border-radius: 40px / 20px;
    border-radius: 40px / 20px;
}

.circle {
    -moz-border-radius: 75px;
    -webkit-border-radius: 75px;
    border-radius: 75px;
}
```

图 14.4.3 四个 border-radius 示例的 CSS 包括了用于支持旧版本的 Firefox 和 Safari 的带厂商前缀的属性。注意 Opera 10.5 和 Internet Explorer 9 不需要使用有前缀的属性。更多信息参见图 14.4.4

Firefox Firefox 3.6	-moz-border-radius-topleft -moz-border-radius-topright -moz-border-radius-bottomleft -moz-border-radius-bottomright
Webkit Safari 3 and 4 Chrome 3	-webkit-border-top-left-radius -webkit-border-top-right-radius -webkit-border-bottom-left-radius -webkit-border-bottom-right-radius
标准的CSS3 语法 Firefox 4 Chrome 4 Safari 5 Internet Explorer 10 Opera 10.5	border-top-left-radius border-top-right-radius border-bottom-left-radius border-bottom-right-radius

图 14.4.4　不同的长形式 border-radius 语法，包括用于支持 Firefox 3.6 和 Webkit（Safari 和 Chrome）的语法，以及被大多数新发布的浏览器采用的官方无前缀语法

2. 为元素创建一个圆角

(1) 输入 -moz-border-radius-topleft: *r*，这里的 *r* 是左上方圆角的半径大小，表示为长度（带单位）。这是用于 Firefox 4.0 之前的版本的非标准的旧语法（参见倒数第二条提示）。如果不介意在这些版本中显示为方角，就可以跳过这一步（参见图 14.4.3）。

(2) 输入 -webkit-border-top-left-radius: *r*，这里的 *r* 使用与第 (1) 步中相同的值。

(3) 输入 border-top-left-radius: *r*，这里的 *r* 是左上方圆角的半径大小，表示为长度（带单位）。这是该属性的标准长形式语法。

注意这些步骤仅演示了如何创建左上方圆角，此外，还可以单独创建其他方位的圆角，步骤如下。

- ❑ 创建右上方圆角：用 top-right 替换第 (2) 步和第 (3) 步中的 top-left。作为可选的一步，用 topright 替换第 (1) 步中的 topleft。
- ❑ 创建右下方圆角：用 bottom-right 替换第 (2) 步和第 (3) 步中的 top-left。作为可选的一步，用 bottomright 替换第 (1) 步中的 topleft。
- ❑ 创建左下方圆角：用 bottom-left 替换第 (2) 步和第 (3) 步中的 top-left。作为可选的一步，用 bottomleft 替换第 (1) 步中的 topleft。

3. 创建椭圆形圆角

输入 -moz-border-radius: *x* / *y*，其中 *x* 是圆角在水平方向上的半径大小，*y* 是圆角在垂直方向上的半径大小，均表示为长度（带单位），参见图 14.4.3。这两个值之间应以一个斜杠分隔。

4. 使用 border-radius 创建圆形

(1) 输入 -moz-border-radius: *r*，这里的 *r* 是元素的半径大小（带长度单位）。要创建圆形，可以使用短形式的语法，*r* 的值应该等于元素高度或宽度的一半，参见图 14.4.3。

(2) 输入 -webkit-border-radius: *r*，这里的 *r* 是元素的半径大小（带长度单位）。这条声明用于增加对基于 Webkit 的旧版本的浏览器（如 Chrome 和 Safari）的支持。

(3) 输入 border-radius: *r*，这里的 *r* 是元素的半径大小（带长度单位）。这是标准的无前缀语法。

提 示　不支持 border-radius 的旧的浏览器仅会以方角呈现元素。

提 示　同 border、margin 和 padding 属性一样，border-radius 也有长短两种形式的语法，使用哪一种取决于是否需要为每个角指定不同的值。

提 示　如果要为元素创建半径相同的四个圆角，可以使用较为简单的 border-radius 简记法，就像设置基本的边框样式属性那样，如图 14.4.5 所示。例如，使用 border-radius: 12px; 会为元素创建四个半径为 12 像素的圆角。

```
div {
    /* 让左上角和右下角的圆角半径为5px,
    → 右上角和左下角的圆角半径为10px */
    border-radius: 5px 10px;
}

div {
    /* 让左上角的圆角半径为5px,
    → 右上角的圆角半径为10px,
    → 右下角的圆角半径为0,
    → 左下角的圆角半径为20px */
    border-radius: 5px 10px 0 20px;
}

div {
    /* 让左上角的圆角半径为20px,
    → 右上角的圆角半径为0,
    → 右下角的圆角半径为0,
    → 左下角的圆角半径为0 */
    border-radius: 20px 0 0 0;
}

div {
    /* 让左上角圆角半径为30px */
    -moz-border-radius-topleft: 30px;
    -webkit-border-top-left-radius: 30px;
    border-top-left-radius: 30px;
}
```

图 14.4.5　以下的例子演示了指定 border-radius 值的一些方式，既有使用一对值的，也有单独指定四个圆角的。使用短形式的语法可以消除对更为复杂的长形式语法的顾虑，尤其是需要支持 Firefox 4.0 之前版本的时候

提 示　border-radius 属性是不继承的。

提 示　尽管使用百分数指定圆角半径是允许的，但通常不推荐这样做，因为有些浏览器根据计算出来的元素大小对此进行处理时会表现出不一致的情况。

提 示　在 Firefox 支持 border-radius 之前，它使用非标准的长形式语法（-moz-border-radius-topleft、-moz-border-radius-topright、-moz-border-radius-bottomleft 和 -moz-border-radius- bottomright）为元素创建单个圆角指定属性（参见图 14.4.4），但从 Firefox 4.0 起，它开始支持推荐的语法（border-top-left-radius、border-top-right-radius、border-bottom-left-radius 和 border-bottom-right-radius）。现在，所有新版本的浏览器都支持无前缀的 border-radius 属性。

提 示　如果你认为编写 border-radius 的 CSS 很乱且单调乏味，也不要担心。Randy Jensen 的 CSS3 Generator（www.css3generator.com）等基于 Web 的服务可以简化创建圆角的工作，省去大量的输入。使用该工具，仅需输入半径大小，就会在示例元素上显示效果，从而可以立即查看是否呈现出需要的效果。更好的一点是，它可以生成全部的 CSS，你只需要将它们复制到样式表中即可。对于其他的 CSS3 属性，它也有类似的功能。这样，工作就简单多了！

14.5　为文本添加阴影

最早，text-shadow 是 CSS2 规范的一部分，接着在 CSS2.1 中被移除了，后来在 CSS3 中又出现了。使用该元素，可以在不使用图像表示文本的情况下，为段落、标题等元素中的文本添加动态的阴影效果（参见图 14.5.1、图 14.5.2 和图 14.5.3）。

```
...
<body>
<h1>Text Shadow</h1>
<h1 class="multiple">Multiple Text Shadows
→ </h1>
</body>
</html>
```

图 14.5.1　演示 text-shadow 的使用的两个示例

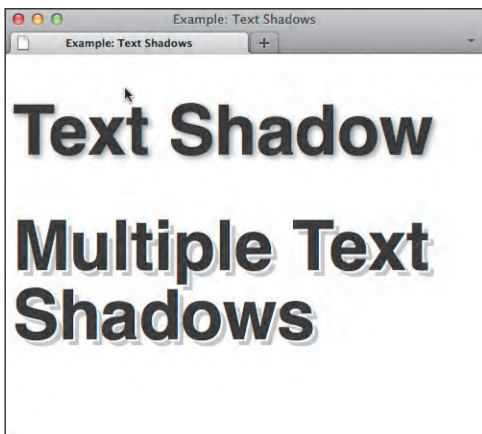


图 14.5.2　两个示例在支持 text-shadow 属性的浏览器中显示的样子

```
h1 {
     font-family: Helvetica, Arial,
sans-serif;
     font-size: 72px;
     line-height: 1em;
     text-shadow: 2px 2px 5px #999;
}

.multiple {
     text-shadow: 2px 2px 0 rgba(255,255,
     → 255,1), 6px 6px 0 rgba(50,50,50,.25);
}
```

图 14.5.3　可以对单个元素添加多个阴影，每条阴影的设置之间用逗号分隔，如应用到 .multiple 类选择器中的声明所示。这样，通过组合使用多个阴影，可以创建出独特有趣的效果

1. 为元素的文本添加阴影

(1) 在 CSS 中，输入 text-shadow:。

(2) 分别输入表示 x-offset（水平偏移量）、y-offset（垂直偏移量）、color（颜色）和 blur radius（模糊半径）的四个值（带长度单位，不用逗号分隔），例如 2px 2px 5px #999（关于允许输入的值，参见提示）。

(3) 输入 ;（分号）。

2. 为元素的文本添加多重阴影

(1) 在 CSS 中，输入 text-shadow:。

(2) 分别输入表示 x-offset、y-offset、color 和 blur radius 的四个值（带长度单位，不用逗号分隔）。关于允许输入的值，参见提示。

(3) 输入 ,（逗号）。

(4) 对四种属性使用不同的值重复第 (2) 步。

(5) 输入 ;（分号）。

3. 将 text-shadow 改回默认值

(1) 在 CSS 中，输入 text-shadow:。

(2) 输入 none。

(3) 输入 ;（分号）。

提示　text-shadow 属性不需要使用厂商前缀。

提示　尽管 text-shadow 的语法与边框和背景属性的语法是类似的，但它不能像边框和背景那样单独地指定四个属性值。

提示　如果不对 text-shadow 的值进行设置，它就会使用初始值 none。

提示　text-shadow 属性是继承的。

提示　text-shadow 属性接受四个值：带长度单位的 x-offset、带长度单位的 y-offset、可选的带长度单位的 blur radius 以及 color 值。如果不指定 blur radius，将假定其值为 0。

提示　x-offset 和 y-offset 值可以是正整数，也可以是负整数，也就是说，1px 和 -1px 都是有效的。blur radius 值必须是正整数。这三个值都可以为 0。

提 示　颜色可以表示为十六进制数、RGB、RGBA 或 HSLA 值（参见 7.4 节的“CSS 颜色”），它可以排列在属性值的第一个或最后一个。

提 示　对单个元素应用多重背景可以产生一些高级效果。要应用多重背景，应使用逗号将不同的阴影属性分隔开，例如 text-shadow: 2px 2px 0 #999, 6px 6px 0 rgba(50, 50, 50, .25);（参见图 14.5.3）。这些阴影将按照倒序进行叠加，第一个显示在最顶层，之后的每一个都位于前一个的下面。

14.6　为其他元素添加阴影

使用 text-shadow 属性可以为元素的文本添加阴影，使用 box-shadow 属性则可以为元素本身添加阴影，如图 14.6.1 和图 14.6.2 所示。它们的基础属性集是相同的，不过 box-shadow 还允许使用两个可选的属性——inset 关键词属性和 spread 属性（用于扩张或收缩阴影）。

```
...
<body>
<div class="shadow">
<h1>Single Shadow</h1>
</div>

<div class="inset-shadow">
<h1>Inset Shadow</h1>
</div>

<div class="multiple">
<h1>Multiple Shadows</h1>
</div>
</body>
</html>
```

图 14.6.1　这个文档包含三个 div，它们用于演示使用 box-shadow 添加一个或多个阴影的效果

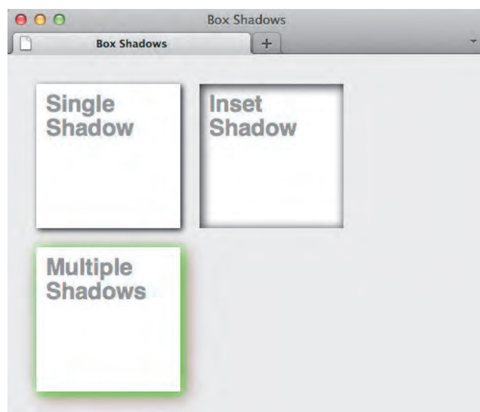


图 14.6.2　三个示例在支持 box-shadow 属性的浏览器中显示的样子

box-shadow 属性与 text-shadow 属性的另一个区别是，前者获得的支持更少，且需要针对某些浏览器版本使用厂商前缀。

尽管通常只对 box-shadow 属性使用四个值，实际上该属性接受六个值：带长度单位的 x-offset 和 y-offset、可选的 inset 关键字、可选的带长度单位的 blur-radius、可选的带长度单位的 spread 值及 color 值。如果不指定 blur-radius 和 spread 的值，则设为 0。

1. 为元素添加阴影

(1) 在 CSS 中，输入 -moz-box-shadow:。

(2) 分别输入表示 x-offset、y-offset、blur radius 和 color 的值（前三个值均带长度单位），如图 14.6.3 所示。

(3) 输入 -webkit-box-shadow:，再重复第 (2) 步。

(4) 输入 box-shadow:，再重复第 (2) 步。

2. 创建内阴影

(1) 在 CSS 中，输入 -moz-box-shadow:。

(2) 在冒号后输入 inset，再输入一个空格。

(3) 分别输入表示 x-offset、y-offset、

blur radius 和 color 的值（前三个值均带长度单位），例如 2px 2px 5px #000。

(4) 输入 -webkit-box-shadow:，再重复第 (2) 步和第 (3) 步。

(5) 输入 box-shadow:，再重复第 (2) 步和第 (3) 步。

```
div {
    background: fff;
    float: left;
    height: 150px;
    margin: 10px;
    width: 150px;
}

.shadow {
    background: #ccc;
    -moz-box-shadow: 2px 2px 5px #000;
    -webkit-box-shadow: 2px 2px 5px #000;
    box-shadow: 2px 2px 5px #000;
}

.inset-shadow {
    -moz-box-shadow: inset 2px 2px 10px
    → #000;
    -webkit-box-shadow: inset 2px 2px
    → 10px #000;
    box-shadow: inset 2px 2px 10px #000;
}

.multiple {
    -moz-box-shadow: 2px 2px 10px
    → rgba(0,255,0,.75), 5px 5px 20px
    → rgba(125,0,0,.5);
    -webkit-box-shadow: 2px 2px 10px
    → rgba(0,255,0,.75), 5px 5px 20px
    → rgba(125,0,0,.5);
    box-shadow: 2px 2px 10px
    → rgba(0,255,0,.75), 5px 5px 20px
    → rgba(125,0,0,.5);
}
```

图 14.6.3　用于创建三个示例的 CSS。注意，要添加两个额外的带厂商前缀的属性，以保证在旧的 Firefox 和 Webkit 浏览器中正确显示。不理解 box-shadow 的浏览器会忽略这些 CSS 规则，页面将显示为没有阴影的效果

3. 为元素应用多重阴影

(1) 在 CSS 中，输入 -moz-box-shadow:。

(2) 分别输入表示 x-offset、y-offset、blur radius 和 color 的值（前三个值均带长度单位），例如 2px 2px 5px #000。

(3) 输入 ,（逗号）。

(4) 对每种属性使用不同的值重复第 (2) 步。

(5) 输入 -webkit-box-shadow:，再重复第 (2) 步至第 (4) 步。

(6) 输入 box-shadow:，再重复第 (2) 步至第 (4) 步。

4. 将 box-shadow 改回默认值

(1) 在 CSS 中，输入 -moz-box-shadow: none。

(2) 输入 -webkit-box-shadow: none。

(3) 输入 box-shadow: none。

提示　对于 box-shadow 属性，Firefox 3.5 和 3.6 需要 -moz- 厂商前缀，一些旧版本的基于 Webkit 的浏览器（如 Safari 和 Chrome）需要 -webkit- 前缀。Opera 10.5 和 Internet Explorer 9 均支持 box-shadow 属性，因此它们不需要厂商前缀。要查找关于什么时候需要对 box-shadow 使用厂商前缀的信息，参见 http://css3please.com。

提示　如果不设置属性值，则使用默认值 none。

提示　box-shadow 属性是不继承的。

提示　颜色可以表示为十六进制数、RGB、RGBA 或 HSLA 值（参见 7.4 节的“CSS 颜色”），它可以排列在属性值的第一个或最后一个。

提示 x-offset和y-offset值可以是正整数，也可以是负整数，也就是说，1px 和 -1px 都是有效的。blur radius 值必须是正整数。这三个值都可以为 0。

提示 对单个元素应用多重背景可以产生一些高级效果。要应用多重背景，应使用逗号将不同的阴影属性分隔开，例如 box-shadow: 2px 2px 0 #999, 6px 6px 0 rgba(50, 50, 50, .25);（参见图 14.6.3）。这些阴影将按照倒序进行叠加，第一个显示在最顶层，之后的每一个都位于前一个的下面。

提示 使用 Internet Explorer 专有的滤镜和 -ms-filter 属性，可以在旧版本的 Internet Explorer 中创建阴影效果，不过，还需要使用额外的 HTML 标记和 CSS 样式规则以解决使用滤镜带来的问题。

14.7 应用多重背景

为单个 HTML 元素指定多个背景是 CSS 中最令人期待的一个特性（如图 14.7.1、图 14.7.2 和图 14.7.3 所示）。通过减少对某些元素的需求（这类元素存在只是为了用 CSS 添加额外的图像背景），指定多重背景便可以简化 HTML 代码，并让它容易理解和维护。多重背景几乎可以应用于任何元素。

```
...
<body>
<div class="night-sky">
    <h1>In the night sky...</h1>
</div>
</body>
</html>
```

图 14.7.1 应用多重背景

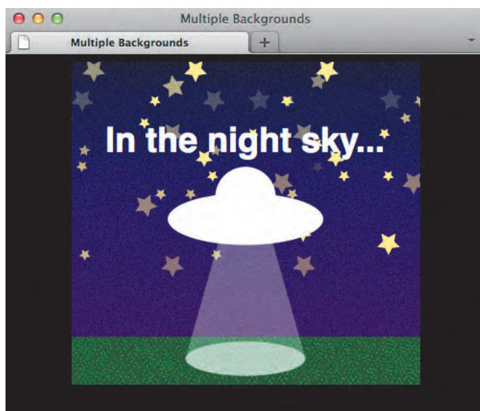


图 14.7.2 支持多重背景的浏览器显示示例的样子。其中，图像是分层次相互重叠在一起的，用逗号分隔的列表中的第一个图像位于顶部。为单个元素应用多重背景是很容易的，不过，为了确保内容是可访问的，需要在 CSS 中使用 background-color 提供一个简单的后备样式

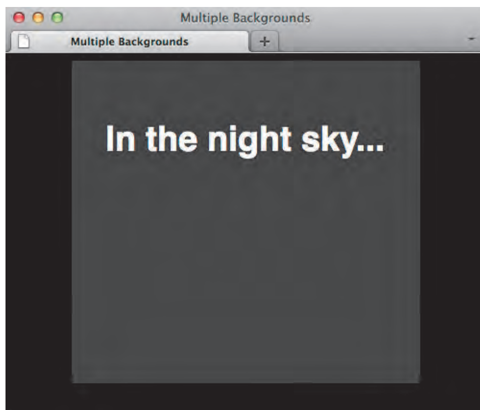


图 14.7.3 这是在不支持多重背景图像语法的浏览器中显示的样子。如果坚持渐进增强的原则，就应该在 background-image 规则之前包含 background-color 或仅包含一个 background-image 属性，以照顾能力较弱的浏览器

为单个元素应用多重背景图像

(1) 输入 background-color: *b*，这里的 *b* 是希望为元素应用的后备背景颜色（参见图 14.7.4）。

14

(2) 输入 background-image: *u*，这里的 *u* 是绝对或相对路径图像引用的列表（用逗号分隔）。

(3) 输入 background-position: *p*，这里的 *p* 是成对的 x-offset 和 y-offset（可以是正的，也可以是负的；带长度单位）的集合（用逗号分隔）。每个背景图像都应有一组 x-offset 和 y-offset。

(4) 输入 background-repeat: *r*，这里的 *r* 是 repeat-x、repeat-y 或 no-repeat 值（参见 10.9 节）的列表（用逗号分隔），每个图像对应一个值。

```
.night-sky {
    background-color: #333;
    background-image: url(ufo.png),
    → url(stars.png), url(stars.png),
    → url(sky.png);
    background-position: 50% 102%, 100%
    → -150px, 0 -150px, 50% 100%;
    background-repeat: no-repeat,
    → no-repeat, no-repeat, repeat-x;
    height: 300px;
    margin: 0 auto;
    padding-top: 36px;
    width: 75%;
}
```

图 14.7.4　要使用多重背景，需要使用四个单独的长格式背景属性：background-color、background-image、background-position 和 background-repeat。使用其中任何一个属性，都可以调整图像的定位和重复方式

提示　指定多重背景不需要使用厂商前缀。

提示　对于多重背景图像，可以使用标准的短形式语法，即使用逗号分隔每组背景参数。例如，使用 background: url(image.jpg) 0 0 no-repeat, url(image2.jpg) 100% 10% no-repeat; 可以实现与重复更多的长形式语法相同的效果。

提示　背景图像是分层次相互叠加在一起的，第一个图像位于顶部，最后一个图像位于底部。

提示　对于不支持多重背景图像的浏览器，如果指定了 background-color 值，它就会作为浏览器所用的全部图像后面的最终背景层。

提示　不支持多重背景图像的浏览器会忽略 background-image 属性，并试着使用后备的 background-color 值。

14.8　使用渐变背景

渐变背景也是 CSS3 中的新特性，通过它可以在不使用图像的情况下创建从一种颜色到另一种颜色的过渡（参见图 14.8.1 和图 14.8.2）。尽管相关的规范仍在变动，但随着规范不断接近于最终的版本，浏览器的支持程度也在不断地提升。

```
<body>
<div class="horizontal"></div>
<div class="vertical"></div>
<div class="diagonal"></div>
<div class="radial"></div>
<div class="multi-stop"></div>
</body>
```

图 14.8.1　仅使用 CSS 实现渐变的五种方式

尽管背景渐变语法需要使用厂商前缀以最大程度地支持各类浏览器，但为了降低学习难度，一开始我们将仅演示无前缀的属性。本节的提示提供了一些额外的信息，完整的示例（包括带厂商前缀的属性）可以在本章的代码下载中找到。

根据渐进增强的原则，最好为不支持背景渐变属性的浏览器提供一个后备选项。这个后备可以是一个简单的背景颜色或背景图像。在 CSS 中，它可以位于背景渐变规则的前面。

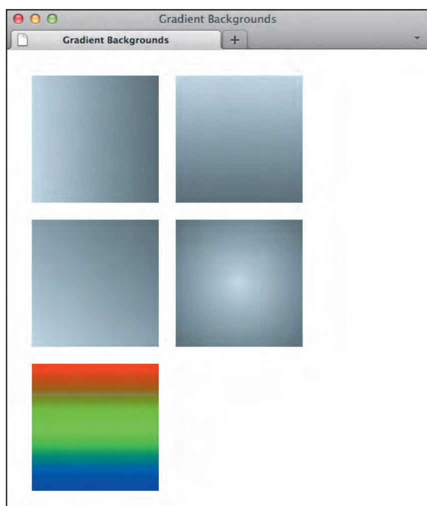


图 14.8.2 支持渐变背景的浏览器显示五个示例的样子（假定已在示例代码中添加了适当的厂商前缀）。对于背景渐变，当前的浏览器均需要使用厂商前缀。如果指定了 background 属性，不理解渐变语法的浏览器将使用该属性的值（另见彩插）

使用 CSS 创建渐变（包括线性渐变和径向渐变）有两种主要的方式，每种方式都有不同的必选参数和可选参数，如图 14.8.3 和图 14.8.4 所示。

```
div {
    float: left;
    height: 150px;
    margin: 10px;
    width: 150px;
}

.horizontal {
    background: #cedce7;
    background: linear-gradient(left,
    ➝ #cedce7,#596a72);
}
```

图 14.8.3 简单的双色水平渐变，使用了标准的线性渐变语法，同时针对不支持 CSS 渐变的浏览器准备了一种简单的后备颜色

```
.vertical {
    background: #cedce7;
    background: linear-gradient(top,
    ➝ #cedce7,#596a72);
}
```

图 14.8.4 创建垂直渐变仅需修改第一个属性，使之包含 top 或 bottom

14

1. 创建后备背景颜色

输入 background-color: *color*，这里的 *color* 可以是任何支持的颜色名称、十六进制数以及 RGB、RGBA 或 HSL 值。

2. 定位渐变类型

输入 background: *type*(，这里的 *type* 可以是 linear-gradient（线性渐变）或 radial-gradient（径向渐变）。

3. 定义渐变开始位置

输入 left 和 ,（逗号）以创建从元素左侧开始的渐变；

或者输入 right 和 ,（逗号）以创建从元素右侧开始的渐变；

或者输入 top 和 ,（逗号）以创建从元素顶端开始的渐变；

或者输入 bottom 和 ,（逗号）以创建从元素底端开始的渐变；

或者输入 angle（角度）值（如 0deg、45deg 或 120deg[1]）和 ,（逗号）以改变渐变的角度。这里的角度为渐变线沿逆时针方向转至水平线所转过的角度，如图 14.8.5 所示；

或者输入 center（仅适用于径向渐变）和 ,（逗号）以创建从元素中心开始的渐变，如图 14.8.6 所示。

① deg 为 degree（度）一词的缩写。——译者注

```
.diagonal {
    background: #cedce7;
    background: linear-gradient(45deg,
    → #cedce7, #596a72);
}
```

图 14.8.5 要创建带角度的渐变，仅需修改第一个属性的值，这个值用于指定渐变的发出角度值。这里的角度为渐变线沿逆时针方向转至水平线所转过的角度，例如，使用 0deg 会创建一个从左至右的水平渐变，使用 90deg 则会创建一个由下至上的垂直渐变

```
.radial {
    background: #cedce7;
    background: radial-gradient(center,
    → #cedce7,#596a72);
}
```

图 14.8.6 径向渐变包含两个额外的可选参数，不过这个最简单的示例则使用了与线性渐变相同的参数。在这个例子中，通过指定 center 关键字，渐变从元素的中心发出

4. 指定开始颜色和结束颜色

输入 *c1*, *c2*)，其中 *c1* 和 *c2* 分别是渐变的开始颜色和结束颜色。使用颜色名称、十六进制数以及 RGB、RGBA 或 HSL 值指定颜色。

5. 创建包含多个颜色的渐变

(1) 重复创建线性渐变或径向渐变的头两项技术，即指定渐变的类型和开始位置（参见图 14.8.2、图 14.8.3 和图 14.8.4）。

(2) 然后输入 *c1 p1*, *c2 p2*, *c3 p3*)，其中 *c#* 表示颜色（使用颜色名称、十六进制数以及 RGB、RGBA 或 HSL 值进行指定），*p#* 是对应颜色的位置（使用 0 到 100% 之间的百分数进行指定），如图 14.8.7 所示。

```
.multi-stop {
    background: url(multi-stop-gradient.jpg)
    → 0 0 repeat-x;
    background: linear-gradient(top, #ff0000
    → 0%, #00ff00 50%, #0000ff 100%);
}
```

图 14.8.7 多步渐变（使用两种以上颜色的渐变）遵循相同的模式，不过还需要指定 color-stop 位置。该位置使用 0 到 100% 之间的百分数表示

提示 基于 Webkit 的浏览器（如 Safari 4）的早期版本使用非标准的语法指定渐变背景。Safari 5 和 Chrome 的新版本均支持与 Firefox 相同的语法，但当前仍需使用 -webkit- 和 -moz- 厂商前缀。

提示 关于使用渐变属性的最新细节信息，参见 Mozilla 和 Webkit 团队提供的介绍（分别位于 https://developer.mozilla.org/en/CSS/-moz-radial-gradient 和 http://://webkit.org/blog/1424/css3-gradients/）。

提示 要创建多色渐变，可以在渐变语法中指定两个以上的颜色，并使用一个额外的可选参数（color-stop），如图 14.8.7 所示。

提示 可以使用颜色名称、十六进制数以及 RGB、RGBA 或 HSLA 值指定颜色。

提示 尽管最新版本的 Web 浏览器对渐变语法的支持程度始终在提升，但渐变语法仍在变化之中，需要使用厂商前缀，包括 Internet Explorer 和 Opera 的厂商前缀。

提示 可用使用 ColorZilla 提供的 Gradient Generator（http://colorzilla.com/gradient-editor/）、Microsoft 的 CSS Gradient Background Maker（http://ie.microsoft.com/testdrive/graphics/cssgradientbackgroundmaker/）等可视化的工具来完成创建 CSS 渐变代码的繁琐工作。这些工具还可以自动生成所有带厂商前缀的属性，从而确保最大程度地兼容旧的浏览器版本。

提示　Internet Explorer 10 包含对 CSS 渐变的原生支持。IE10 之前的版本则可以使用专有的 filter:progid:DXImageTransform.Microsoft.gradient 滤镜创建渐变，或者使用额外的 HTML 标记和 SVG（针对 Internet Explorer 9）。像上一条提示中提到的 ColorZilla Gradient Editor 这样的工具可以生成所有使用滤镜所需的代码，因此你不必担心需要自己编写这些代码。

提示　可以通过指定 background-color 或 background-image 为不支持渐变的浏览器提供后备方案，不过要记住的是，无论浏览器是否使用 CSS 中的图像，这些图像都会被下载。

提示　此外，Firefox 和 Webkit 还支持带前缀的 repeating-linear-gradient（重复线性渐变）和 repeating-radial-gradient（重复径向渐变）。

14.9　为元素设置不透明度

使用 opacity 属性可以修改元素（包括图像）的透明度，如图 14.9.1 和图 14.9.2 所示。

修改元素不透明度的方法：

输入 opacity: *o*，这里的 *o* 表示元素的不透明程度（两位小数，不带单位）。

```
…
<body>
<div class="box">
    <img src="sleeves.jpg" width="420"
height="296" alt="Record Sleeves" />
</div>
</body>
</html>
```

图 14.9.1　这个文档包含一个 div，其中有一个图像

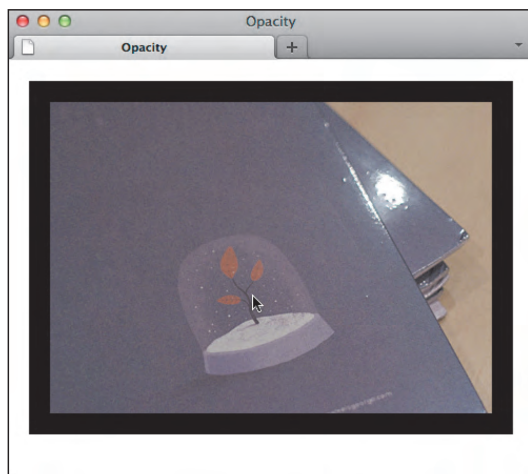


图 14.9.2　这是示例在 div 元素的 opacity 设为默认值 1 时的样子（另见彩插）

提示　opacity 的默认值为 1。该属性值可以使用 0.00（完全透明）至 1.00（完全不透明）之间的两位小数，如图 14.9.3 和图 14.9.4 所示。

提示　通过使用 opacity 属性和 :hover 伪属性，可以产生一些有趣且实用的效果。例如，可以修改元素在用户鼠标停留时的不透明度，或者为某些元素（如可选的表单字段）添加表示禁用的外观。

```
img {
    vertical-align: top;
}

.box {
    background: #000;
    opacity: .5;
    padding: 20px;
}
```

图 14.9.3　将 opacity 设为小于 1 的值可以让元素及其子元素变成半透明的样子。在这个例子中，不透明度被设为 50% 或 0.5，小数点前的 0 是可选的

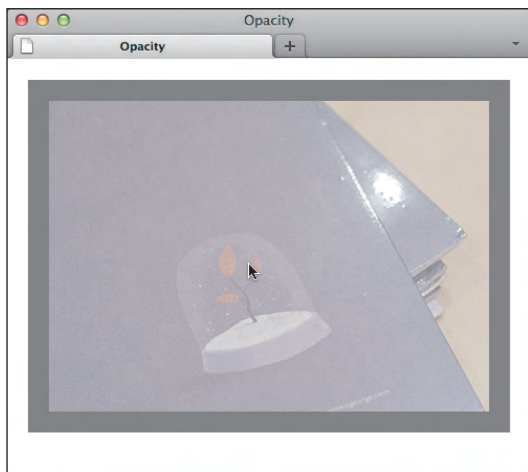


图 14.9.4 这是示例在 div 元素的 opacity 设为 0.5（即不透明度为 50%）时的样子。可以看到，div 元素的黑色背景现在显示为灰色，图像也显示为半透明的效果（另见彩插）

提示 就像在 RGBA 和 HSLA 颜色中设置 opacity 值一样，设置 opacity 属性值时不需要在小数点前加上 0。

提示 无论显示成什么样子，opacity 属性并不是继承的。opacity 的值小于 1 的元素的子元素也会受到影响，但这些子元素的 opacity 值仍为 1。

提示 Internet Explorer 9 之前的版本并不原生支持 opacity 属性，但对于它们，可以使用专有的 -ms-filter:progid:DXImageTransform.Microsoft.Alpha(opacity=50); 和 filter:alpha(opacity=50); 属性，并为元素设置 zoom: 1; 以触发 hasLayout，如图 14.9.5 所示。关于这些滤镜，可以在 CSS Tricks 网站上找到更多信息（http://css-tricks.com/css-transparency-settings-for-all-browsers/）。关于 hasLayout 的出处和作用，参见 http://haslayout.net/haslayout。

```
div {
    /* 将元素的不透明度设为50%，为Internet Expl-
    → orer  9之前的版本提供可选的专有滤镜声
    → 明，并使用zoom: 1声明为这些IE的旧版本触
    → 发hasLayout */
    -ms-filter: progid:DXImageTransform.
    → Microsoft.Alpha(opacity=50);
    filter: alpha(opacity=50);
    opacity: .5;
    zoom: 1;
}
```

图 14.9.5 这个简单的示例演示了如何为并不原生支持 opacity 的 Internet Explorer 9 之前的版本应用其专有滤镜。-ms-filter: 声明是针对 IE8 的，而更为简单的 filter: 则支持 IE5 至 IE7

第 15 章

列　　表

15

本章内容

- ❑ 创建有序列表和无序列表
- ❑ 选择标识
- ❑ 选择列表的起始编号
- ❑ 使用定制的标识
- ❑ 控制标识的位置
- ❑ 同时设置所有的列表样式属性
- ❑ 设置嵌套列表的样式
- ❑ 创建描述列表

HTML 包含专门用于创建项目列表的元素。你可以创建普通列表、编号列表、符号列表以及描述列表，可以在一个列表中嵌套另外一个或多个列表。

所有的列表都是由主要元素和次要元素构成的。主要元素用于指定要创建的列表的类型，其中，ul 表示无序列表（unordered list），ol 表示有序列表（ordered list），dl 表示描述列表（description list；在 HTML5 之前称为定义列表）。次要元素用于指定要创建的列表项目类型，其中，li 代表 ol 或 ul 中的列表项目，dt 和 dd 分别代表 dl 中的术语和描述。

在这些类型里面，无序列表是万维网上最为常见的列表类型，它也是对大多数类型的导航进行标记的事实标准（本书有几个这样的例子）。不过，上述三种类型都各有用武之地，本章将对此进行讲解。

15.1　创建有序列表和无序列表

有序列表适于提供完成某一任务的分步说明（如图 15.1.1 和图 15.1.2 所示），或用于创建大型文档的大纲（如果愿意，可以加上指向对应部分的链接）。它还适合对面包屑导航进行标记（参见提示）。总之，它适用于任何强调顺序的项目列表。

无序列表可能是万维网上使用范围最广的列表，因为它们常用于标记导航（如图 15.1.3 和图 15.1.4 所示）。

创建列表

(1) 输入 （有序列表）或 （无序列表）。对于有序列表，可以包含 start、type 和 reversed 这三个可选的属性。（关于 start，参见 15.3 节；关于 type，参见 15.2 节；关于 reversed，参见最后一条提示。reversed 还没有浏览器支持，因此无法看到效果。）

(2) 输入 （这是 list 一词的前两个字母）以开始第一个列表项目。对于有序列表，可以包含可选的 value 属性（更多细节参见 15.3 节）。

(3) 添加要包含在列表项目内的内容（如文本、链接、img 元素等）。

(4) 输入 以结束列表项目。

(5) 对于每个新的列表项目，重复第 (2) 步至第 (4) 步。

(6) 输入 或 （与第 (1) 步中的开始标记对应）以结束列表。

```
...
<body>
<h1>Changing a light bulb</h1>
<ol>
    <li>Make sure you have unplugged the
    →lamp from the wall socket.</li>
    <li>Unscrew the old bulb.</li>
    <li>Get the new bulb out of the
    →package.</li>
    <li>Check the wattage to make sure
    →it's correct.</li>
    <li>Screw in the new bulb.</li>
    <li>Plug in the lamp and turn it
    →on!</li>
</ol>
</body>
</html>
```

图 15.1.1　目前还没有对列表标题进行格式化的正式方法。大多数情况下，使用常规的标题（参见第 3 章）或段落（参见第 4 章）即可，就像下面的例子那样。按照惯例（并非必须），可以对列表项目进行缩进，表明它们是嵌套在 ol 里面的（对于 ul 也是这样的）。不过，这些缩进在页面中不会显示出来，页面中的显示是由应用到列表上的 CSS 控制的



图 15.1.2　这个列表使用默认的阿拉伯数字创建带编号的有序列表。可以使用 CSS 对此进行修改。有序列表和无序列表在显示时默认都会进行缩进，无论它们在 HTML 中是否有缩进（参见图 15.1.1）

```
...
<body>
<h1>PageWhacker, version 12.0: Features</h1>
<ul>
    <li>New or improved features marked
    →with a solid bullet.</li>
    <li>One-click page layout</li>
    <li>Spell checker for 327 major
    →languages</li>
    <li>Image retouching plug-in</li>
    <li>Special HTML filters</li>
    <li>Unlimited Undo's and Redo's</li>
    <li>Automatic book writing</li>
</ul>
</body>
</html>
```

图 15.1.3　无序列表中的项目与有序列表中的是相同的，只有 ul 元素是不同的



图 15.1.4　在默认情况下，无序列表前面显示实心的圆点。可以通过 CSS 对此进行修改

提 示　不要根据希望添加在内容旁边的标识样式决定要使用的列表类型，毕竟，这些标识的样式随时都可以通过 CSS 进行修改（甚至可以在有序列表上显示符号）。相反，应该考虑列表的含义——列表会随着其中项目顺序的改变而改变吗？如果答案是肯定的，就使用有序列表进行标记。否则，就使用无序列表。

提示 使用列表标记链接组时，大多数情况下均可以使用无序列表，如主导航链接、指向一组视频或相关报道的链接、页脚中的一组链接等。对于面包屑导航则应使用有序列表，因为这些链接有特定的次序（换句话说，顺序是有意义的）。面包屑导航通常水平地显示在主要内容区域的上方，指示当前页面在站点导航路径中的位置。例如，在提供某款移动电话详细信息的页面，面包屑呈现为 Home → Products → Phones → The Fone 3.0。列表中的每个项目均为链接（最后一个项目除外，因为访问者正位于 The Fone 3.0 的页面）。图 15.7.5 包含了这个例子（面包屑导航位于顶部主导航和产品名称的大标题之间）。

提示 第 11 章的完整示例网页演示了各种不同的使用和呈现列表的方式。其中有用于导航的无序列表（水平排列，有符号）、用于一系列图像的无序列表（水平排列，无标识）以及用于一系列按时间排序的月度存档链接的有序列表（有符号）。第 3 章也有在导航中使用 ul 的例子。

提示 除非使用 CSS 另行指定，有序列表中的项目使用阿拉伯数字（1、2、3 等）进行编号（参见图 15.1.1）。

提示 默认情况下，无序列表的项目前面显示实心的圆点（参见图 15.1.2）。你可以选择不同的符号（参见 15.2 节），甚至可以创建你自己的符号（参见 15.4 节）。

提示 只能将列表内容放在 li 元素里。例如，将内容放置在 ol 或 ul 开始标记和第一个 li 元素之间是不允许的。

提示 可以在 li 元素中嵌套各种类型的 HTML 元素，如任何短语内容元素（如 em、a、cite 等）。在列表项目中嵌套段落和 div 也是有效的，但很少需要这样做。

提示 可以在一个列表中创建另一个列表（即**嵌套列表**，nesting list），甚至可以混合使用有序列表和无序列表。不过，要确保用到所有需要的开始标记和结束标记，对每个列表进行正确地嵌套。嵌套有序列表和无序列表的例子见 15.7 节。

提示 默认情况下，列表有一定的左侧外边距，从而形成缩进。可以使用 CSS 取消（或增大）缩进。根据减少左侧外边距的量，符号可能处于内容的外边或者消失在窗口的左边缘之外（第 11 章中有这样的例子）。

提示 如果将内容方向指定为从右向左（例如在语言为希伯来语的情况下），列表就会通过右侧外边距进行缩进。要指定内容方向，可以对页面的 html 元素中设置 dir 属性，如 <html **dir="rtl"** lang="he">。在这个例子中，lang 被设为表示希伯来语的 he。还可以在 body 里面的元素上设置 dir 和 lang 以覆盖 html 元素上的设置。dir 属性的默认值为 ltr。

提示 截至本书写作之际，还没有浏览器支持布尔型的 reversed 属性。该属性的作用是指示降序排列的有序列表（使用 <ol **reversed**> 或 <ol **reversed="reversed"**> 均可指定该属性）。

15.2 选择标识

创建列表时，无论是有序列表（如图 15.2.1 所示）还是无序列表，都可以选择

出现在列表项目左侧的标识的类型（如符号、编号、图像等）。

```
...
<body>
<h1>The Great American Novel</h1>
<ol>
    <li>Introduction</li>
    <li>Development</li>
    <li>Climax</li>
    <li>Denouement</li>
    <li>Epilogue</li>
</ol>
</body>
</html>
```

图 15.2.1　这是用做示例的有序列表。我们将对其应用大写罗马数字（upper-roman）

1. 选择标识

在样式表中，输入 list-style-type: *marker*，这里的 *marker* 是以下属性值中的一种。

- disc（圆点，●）
- circle（圆圈，○）
- square（方块，■）
- decimal（数字，1、2、3……）
- upper-alpha（大写字母，A、B、C……）
- lower-alpha（小写字母，a、b、c……）
- upper-roman（大写罗马数字，Ⅰ、Ⅱ、Ⅲ、Ⅳ……），如图 15.2.2 和图 15.2.3 所示
- lower-roman（小写罗马数字，i、ii、iii、iv……）

```
li {
    list-style-type: upper-roman;
}
```

图 15.2.2　可以对任何列表项目应用 list-style-type 属性。如果这个页面上有两个列表，其中一个是无序列表，那么可以将这个例子中的选择器改为 **ol li**，从而仅对有序列表应用大写罗马数字

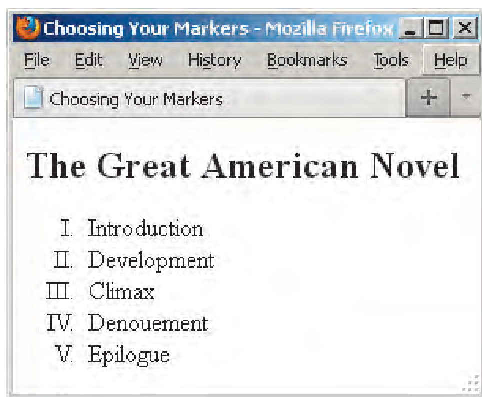


图 15.2.3　现在，有序列表采用大写罗马数字进行编号。注意，大多数浏览器会让编号右对齐（不过列表项目的内容是左对齐的，如图所示）

2. 显示无标识列表

在样式表规则中，输入 list-style-type: none。

提示　默认情况下，对于无序列表，第一级列表使用圆点，第一级嵌套列表（第二级列表）使用圆圈，第三级及后续各级列表使用方块。参见 15.7 节。

提示　在不同的浏览器中，圆点、圆圈和方块符号的大小和外观可能稍有差异。

提示　可以通过 list-style-type 对 ol 和 ul 设置任何标识样式。换句话说，可以对 ol 使用方块标识，也可以对 ul 使用数字标识。

提示　也可以在 HTML 中通过 type 属性为有序列表指定标识类型，尽管推荐的做法是尽可能在 CSS 中定义列表样式类型。type 属性可接受的值包括 A、a、I、i 和 1，它们用于指定要使用的编号类型（默认值为 1）。例如，**<ol type="I">** 表示使用大写罗马数字作为编号。

15.3　选择列表的起始编号

你可能希望列表的编号从默认值 1 以外的某个数字开始，如图 15.3.1 和图 15.3.2 所示。

```
...
<body>
<h1>Changing a light bulb (with a few steps
→ missing)</h1>
<ol start="2">
     <li>Unscrew the old bulb.</li>
     <li value="5">Screw in the new bulb.
     → </li>
     <li>Plug in the lamp and turn it on!
     → </li>
</ol>
</body>
</html>
```

图 15.3.1　在这个例子中，我略去了列表中的一些步骤，不过仍希望余下的步骤保持原有的编号。因此，整个列表从 2 开始编号（使用 start="2"），并将第二个项目的值设为 5（使用 value="5"）。这两个属性都是可选的，也不一定要像这样同时使用

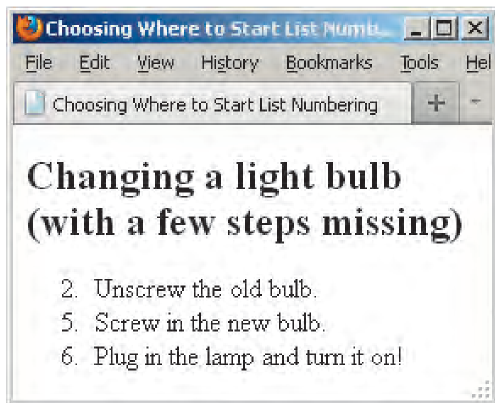


图 15.3.2　注意，不仅第一个和第二个项目是按我们指定的规则进行编号的，连第三个项目（Plug in the lamp and turn it on!）也受到了影响

1. 设置整个列表编号方案的初始值

在 ol 开始标记中输入 start="*n*"，这里的 *n* 表示列表的初始值。

2. 修改有序列表中某列表项目的编号

在目标 li 项目内输入 value="*n*"，这里的 *n* 代表该列表项目的值。value 应该总是使用数字表示，浏览器会自动将它转换为 CSS 或 type 属性中指定的标识类型（参见 15.2 节）。

提示　设置 start 值的时候，应当始终使用数字。即便决定使用字母或罗马数字对列表进行编号（参见 15.2 节），也应使用数字。浏览器会显示预期的标识。

提示　value 属性的值会覆盖 start 属性的值。

提示　使用 value 属性对某列表项目的编号进行修改后，后续的列表项目也会相应地重新编号。

提示　如果需要在有序列表中指定两个或两个以上位置相同的项目，使用 value 是非常方便的。以表示公路赛前五名选手的列表为例，通常该列表会显示为 1、2、3、4、5，但如果有并列第 2 名，则可以将第三个项目写成 **<li value="2">**，这时，列表将显示为 1、2、2、3、4。

提示　列表可以包含一个以上的带 value 属性的 li。

15.4　使用定制的标识

如果对使用圆圈、方块、圆点以及罗马数字这些标识感到厌倦，也可以使用图像创建自己定制的标识。要使用定制的标识，不必修改 HTML（如图 15.4.1 所示），仅需修改 CSS（如图 15.4.2 和图 15.4.3 所示）。

```
...
<body>

<h1>PageWhacker, version 12.0: Features</h1>
<ul>
    <li>One click page layout</li>
    <li>Spell checker for 327 major
    → languages</li>
    <li>Image retouching plug-in</li>
    <li>Unlimited Undo's and Redo's</li>
    <li>Automatic book writing</li>
</ul>
</body>
</html>
```

图 15.4.1　这个列表与任何普通无序列表都是相似的，不过，加上一些 CSS（如图 15.4.2 所示）后，它看起来就不一样了（如图 15.4.3 所示）

```
ul {
    /* 取消默认标识 */
    list-style: none;

    /* 为列表项目设置缩进 */
    margin-left: 0;
    padding-left: 15px;
}

li {
    /* 让图像显示在距离顶端2像素的位置 */
    background: url(arrow-right.png)
    → no-repeat 0 2px;

    /* 将文本向右挤压，为箭头腾出空间 */
    padding-left: 25px;
}
```

图 15.4.2　首先，清除默认的标识（这样就看不到符号和箭头了），调整列表项目的缩进大小。然后为每个列表项目指定箭头背景图像，将它的位置设在距离 li 的顶端几个像素的地方，同时为 li 增加一些左侧内边距，确保文本不会覆盖箭头。要确保 background 的 url 部分中的图像路径是正确的。这里的 url 应该是图像相对于样式表的位置，而不是相对于 HTML 的位置（有关信息参见 10.9 节）



图 15.4.3　默认的圆点被箭头图像取代了

使用定制的标识

(1) 在目标列表或列表项的样式规则中，输入 list-style: none; 以取消常规的标识。

(2) 在目标列表的样式规则中，设置 margin-left 和 / 或 padding-left 属性，指定列表项目缩进的大小。（为了在不同的浏览器上实现相似的效果，通常需要同时设置这两个属性。）同时设置 margin-left: 0; 和 padding-left: 0; 就将取消所有的缩进。注意，如果为内容设置了 dir="rtl"，那么就应该设置 margin-right 和 padding-right 属性。关于在列表中使用 dir、lang 及从右至左的语言，参见 15.1 节的提示。

(3) 在目标列表的 li 元素的样式规则中，输入 background: url(*image.ext*) *repeat-type horizontal vertical*;，其中 *image.ext* 是要作为定制标识的图像的路径和文件名，*repeat-type* 是 no-repeat、repeat-x 和 repeat-y 中的一种（通常设为 no-repeat），*horizontal* 和 *vertical* 值表示列表项目中背景图像的位置（如图 15.4.2 所示）。

输入 padding-left: *value*;，这里的 *value* 应不小于背景图像的宽度，以防列表项目的内容覆盖到定制标识的上面。

提 示　在 url 和前括号之间不应有空格（如图 15.4.2 所示）。包围 URL 的引号是可选的。

提 示　注意，相对 URL 是相对于样式表的，而不是相对于网页的。

提 示　如果想为列表中的某一个或几个项目应用定制的标识，可以为其添加一个类，再为这个类定义样式规则。根据外观要求，可能需要对这些 li 的左侧外边距进行调整，而不是对父元素 ol 或 ul 的外边距和内边距进行调整。

提 示　另一种显示定制标识的方法是使用 list-style-image 属性。例如，li { list-style-image: url(image.png); }。不过，使用这种方法很难达到预期的效果，因为不同浏览器对它的显示效果并不一致。而且，使用图 15.4.2 所示的方法可以更好地控制图像标识的位置，这也是人们倾向于使用这种方法的原因。list-style-image 属性会覆盖 list-style-type 属性。不过，如果出于某种原因导致图像未能加载，则会使用由 list-style-type 指定的标识。

15.5　控制标识的位置

默认情况下，列表会从（其父元素的）左侧外边距进行缩进。标识可以向右对齐距离起点一半长度的位置（这是默认情况），如图 15.5.1 所示，也可以缩在文本块内（称为缩排），如图 15.5.2 和图 15.5.3 所示。

控制标识位置的步骤

(1) 在目标列表或列表项目的样式表规则中，输入 list-style-position:。

(2) 输入 inside 让标识缩在文本块内（参见图 15.5.2），或者输入 outside 让标识显示在列表项目文本的左边（这是默认的设置）。

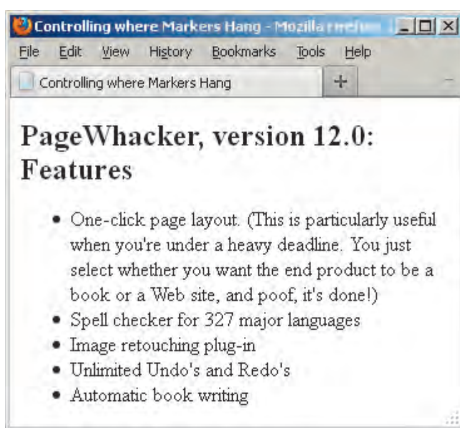


图 15.5.1　这说明了在默认情况下浏览器呈现列表项目文本及标识之间相对位置的方式。标识位于内容的外边。可以通过 CSS 对此进行修改。我在第一个特性中增加了一些文本，从而让标识的悬挂缩进效果更为明显（参见图 15.5.2 和图 15.5.3）

```
li {
    list-style-position: inside;
}
```

图 15.5.2　将 list-style-position 设置为 inside 可以改变显示方式

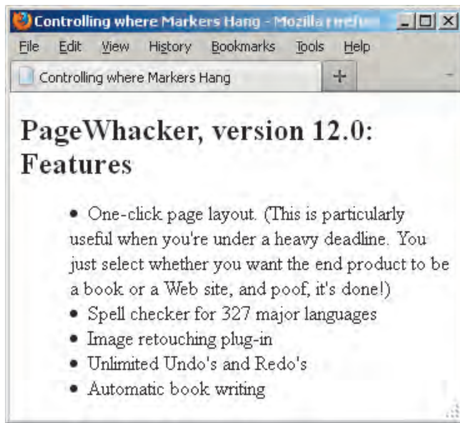


图 15.5.3　每行的标识都是从列表项目的左侧外边距开始的，而不是出现在左侧的外边

15

提示　默认情况下，标识出现在列表段落的外边。

提示　list-style-position 属性是继承的。

15.6　同时设置所有的列表样式属性

同 background、border、font、outline 等属性一样，CSS 也为 list-style 提供了简写形式（如图 15.6.1 和图 15.6.2 所示）。

```
li {
    list-style: inside circle;
}
```

图 15.6.1　使用这条样式规则等价于将 list-style-position 设置为 inside，且将 list-style-type 设为 circle。这样写更为简洁。还可以在简记法中指定 list-style-image（参见第一条提示）

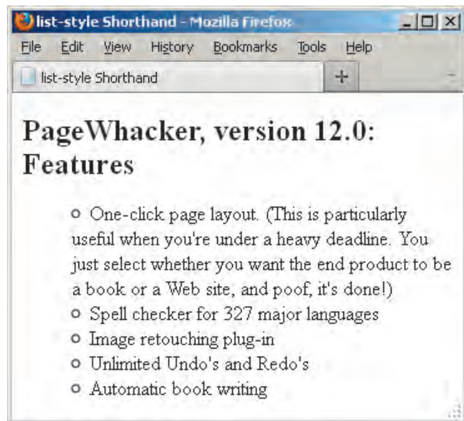


图 15.6.2　代码的实现结果与图 15.5.3 相同，除了将标识改成圆圈以外

同时设置所有的列表样式属性的步骤

(1) 输入 list-style:。

(2) 如果需要，指定应出现在列表项目旁边的标识类型（参见 15.2 节）。

(3) 如果需要，指定列表项目所用的定制标识（参见 15.4 节最后一条提示）。

(4) 如果需要，指定标识应悬挂在列表段落之外或缩在文本块内（参见 15.5 节）。

提示　如果要在简写形式的属性中指定 list-style-image，图 15.6.1 中的例子就应写成 li { list-style: url (arrow-right.png) inside square; }。

提示　可以指定三个 list-style 属性，也可以指定其中的任意一个。图 15.6.1 中指定了两个。

提示　你可能认为，忽略这三个属性中的一个意味着对该属性没有影响，但事实并非如此。任何没有显式地指定的属性都会设为其默认值（list-style-type 的默认值为 disc，list-style-image 的默认值为 none，list-style-position 的默认值为 outside）。

提示　可以按任意次序指定这些属性。

提示　list-style 属性是继承的。

15.7　设置嵌套列表的样式

可以在一个列表中插入另一个列表，里面的列表就称为嵌套列表。对于有序列表和无序列表均可创建嵌套列表（混合在一起或单独嵌套）。此外，还有另外一种嵌套列表，有关示例参见 15.8 节。

如果要按有序列表的结构创建大纲（可能需要好几级的列表项目，如图 15.7.1、图 15.7.2 和图 15.7.3 所示），或者需要按无序列表的结构创建带子菜单的导航（如图 15.7.4 和图 15.7.5 所示，更多细节参见“使用嵌套

列表创建下拉式导航”），就会发现嵌套列表很有用。可以通过很多方式为嵌套列表设置样式，就像示例所展示的那样。

```
...
<body>

<h1>The Great American Novel</h1>
<ol>
    <li>Introduction
       <ol>
          <li>Boy's childhood</li>
          <li>Girl's childhood</li>
       </ol>
    </li>
    <li>Development
       <ol>
          <li>Boy meets Girl</li>
          <li>Boy and Girl fall in love
          ➝ </li>
          <li>Boy and Girl have fight
          ➝ </li>
       </ol>
    </li>
    <li>Climax
       <ol>
          <li>Boy gives Girl ultimatum
             <ol>
                <li>Girl can't believe
                ➝ her ears</li>
                <li>Boy is indignant at
                ➝ Girl's indignance</li>
             </ol>
          </li>
          <li>Girl tells Boy to get
          ➝ lost</li>
       </ol>
    </li>
    <li>Denouement</li>
    <li>Epilogue</li>
</ol>

</body>
</html>
```

图 15.7.1　这里有四个嵌套列表，一个位于 Introduction 列表项目内，一个位于 Development 列表项目内，一个位于 Climax 列表项目内，还有一个突出显示的粗体字列表位于 Boy gives Girl ultimatum 列表项目内（该列表项目又位于 Climax 列表项目内）

```
ol li {
    font-size: .75em;
    list-style-type: upper-roman;
}

ol ol li {
    list-style-type: upper-alpha;
}

ol ol ol li {
    list-style-type: decimal;
}

li li {
    font-size: 1em;
}
```

图 15.7.2　可以对每一级嵌套列表单独进行格式化。如果要对列表中的文本使用 em 或百分数设置字体大小，就需要添加 li li {font-size: 1em; }（或者将 1em 替换为 100%），以防止嵌套列表不断缩小而导致难以辨认（参见最后一条提示）

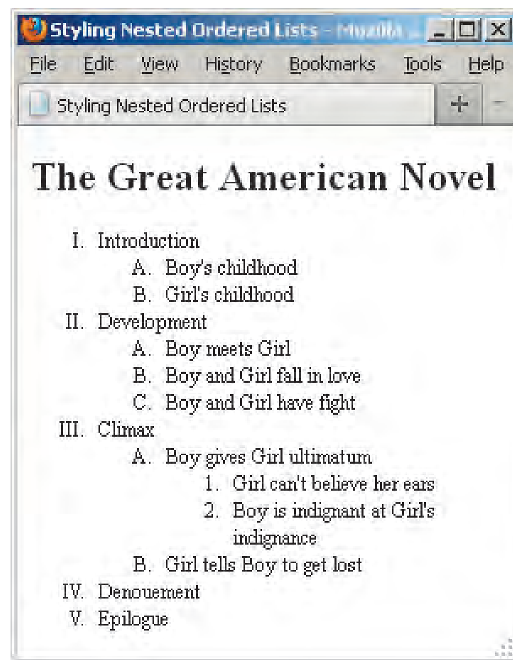


图 15.7.3　第一级列表（ol li）使用大写罗马数字，第二级列表（ol ol li）使用大写字母，第三级列表（ol ol ol li）使用阿拉伯数字

```
...
<body>
<nav role="navigation">
    <ul class="nav">
       <li><a href="/">Home</a></li>
       <li><a href="/products/">Products</a>
          <ul>
             <li><a href="/products/phones.html">Phones</a></li>
             <li><a href="/products/accessories.html">Accessories</a></li>
          </ul>
       </li>
       <li><a href="/support/">Support</a>
          <ul>
             <li><a href="http://www.thephoneycompany.com/support/forum/">Community Forum
             ➝ </a></li>
             <li><a href="/support/contact-us.html">Contact Us</a></li>
             <li><a href="/support/how-to-guides.html">How-to Guides</a></li>
          </ul>
       </li>
       <li><a href="/about-us/">About Us</a></li>
    </ul>
</nav>
...
</body>
</html>
```

图 15.7.4　这是另一个嵌套列表的例子。在这个例子中，使用无序列表构建导航菜单，同时使用两个嵌套的无序列表构建子菜单。注意，每个嵌套的 ul 都包含在其父元素开始标记 <li> 和结束标记 之内。通过一些 CSS，可以让导航水平排列，同时让子菜单在默认情况下隐藏起来，并在访问者激活它们时显示出来，如图 15.7.5 所示

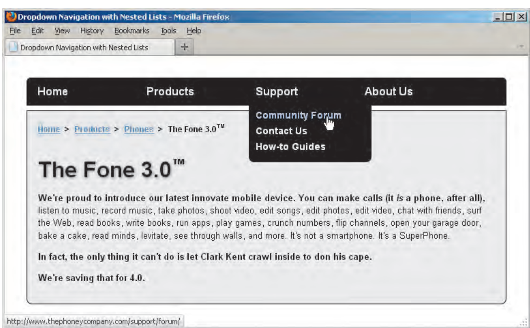


图 15.7.5　Products 和 Support 列表项目都包含嵌套在 ul 中的子菜单，不过，由于应用了一些 CSS，这两个子菜单在默认情况下并不显示出来。在这个例子中， Support 的子菜单显示出来了，这是因为鼠标停留在包含 Support 链接及子菜单嵌套列表的 li 上的缘故。完整的 CSS 见本书配套网站

为嵌套列表设置样式的步骤

(1) 要设置最外层列表的样式，输入 *toplevel li* {*style_rules*}，其中，*toplevel* 是最外层列表的类型（如 ol、ul 或 dt），*style_rules* 是要应用的样式。

(2) 对于第二级列表，输入 *toplevel 2ndlevel li* {style_rules}，其中，*toplevel* 对应于第 (1) 步中的 *toplevel*，*2ndlevel* 则是第二级列表的类型。

(3) 对于第三级列表，输入 *toplevel 2ndlevel 3rdlevel* li {style_rules}，其中，*toplevel* 和 *2ndlevel* 对应于第 (1) 步和第 (2) 步中的 *toplevel* 和 *2ndlevel*，*3rdlevel* 则是第三级列表的类型。

(4) 对于要设置样式的每个嵌套列表，继续以上述方法进行设置。

提 示 选择器应反映文档中嵌套列表的类型，例如，可能需要使用ul ul ol li这样的选择器。

提 示 除了上述方法以外，也可以为每个嵌套列表添加类名，再设置对应的样式。不过，这里演示的方法可以在不改变 HTML 的情况下控制样式。

提 示 无论嵌套的层级是哪一级，有序列表在默认情况下总是使用阿拉伯数字（1、2、3……）。可以使用 list-style-type 指定其他的编号方案。根据 *The Chicago Manual of Style* 一书的建议，正确的列表嵌套次序为 I、A、1、a（此后交替使用 1 和 a 编号方案）。

提 示 默认情况下，对于无序列表，第一级列表使用圆点符号，下一级列表使用空心圆点符号，第三级及后续各级列表使用方块符号。类似地，可以使用 list-style-type 指定希望使用的符号类型（参见 15.2 节）。

提 示 由于列表项目（li 元素）可以嵌套在其他列表项目内，因此在使用相对值指定字体大小时需要留心。如果使用像 li {font-size: .75em; } 这样的样式规则，那么最外层列表项目的字体大小便是其父元素的 75%，因此如果该父元素的字体大小为默认的 16 像素，那么最外层列表项目就是 12 像素大小，这还算好。但是，第一个**嵌套**列表项目的字体大小会是**其**父元素（第一个列表项目，12 像素大小）的 75%，因此只有 9 像素。每一级都会迅速让情况变得更糟糕。一种解决方案是添加 li li {font-size: 1em; }，如图 15.7.2 所示（或者将 1em 替换为 100%）。这样，嵌套列表项目就总是与顶级列表项目一样大，如图 15.7.3 所示。（由 Eric Meyer 提供，www.meyerweb.com。）

15

使用嵌套列表创建下拉式导航

嵌套列表的一种使用场合是创建下拉式（或飞出式）导航菜单（如图 15.7.4 所示）。通过使用 CSS 为导航添加一些样式，可以让每个子菜单仅在访问者鼠标停留在其父元素的列表项目上时显示出来（如图 15.7.5 所示），并在访问者将鼠标移走时隐藏。

可以通过多种方法实现这一效果，但这些方法总离不开在选择器中使用 :hover 伪类以显示子菜单。下面演示了这样一种在默认情况下隐藏嵌套列表，并在访问者鼠标停留时显示该列表的方法：

```
/* 子菜单的默认状态 */
.nav li ul {
  left: -9999em; /* 将子菜单移出屏幕 */
```

```
        position: absolute;
        z-index: 1000;
    }

    /* 当鼠标停留在父元素li上时子菜单的状态 */
    .nav li:hover ul {
        display: block; /* 针对IE的旧版本 */
        left: auto; /* 让子菜单回到自然状态 */
    }
```

对应的 HTML 如图 15.7.4 所示。要实现水平排布，移除列表项目的符号，以及对外观进行调整，都需要使用更多的 CSS。图 15.7.5 中所示页面的完整 HTML 和 CSS 代码见本书配套网站（www.bruceontheloose. com/htmlcss/ examples/chapter-15/dropdown-nav.html）。我还在代码中添加了一些注释。

可以使用类似的方法创建带飞出式子菜单的垂直导航（子菜单显示在导航的侧边）。

15.8　创建描述列表

HTML 提供了专门用于描述成组出现的名称（术语）及其值之间关联的列表类型。这种类型在 HTML5 中称为描述列表（description list），而在 HTML 的早期版本中则称为定义列表。

根据 HTML5 规范，“由名称及其值构成的组合可以是术语和定义、元数据主题和值、问题和答案，以及任何其他的名 – 值组。”每个列表都包含在 dl 中，其中的每个名 – 值组都有一个或多个 dt 元素（名称或术语）以及一个或多个 dd 元素（它们的值）。图 15.8.1 展示了一个基本的描述列表示例。除了使用一个简单的样式规则（如图 15.8.2 所示）应用粗体以外，所有的样式都是默认的（如图 15.8.3 所示）。

以下都是 dl 元素内的 dt 和 dd 元素的组合，这些安排都是有效的。

- 一个 dt 同一个 dd（参见图 15.8.1，同时参见图 15.8.7 中 Cast 下嵌套描述列表中的 Director）。这是最常见的情形。
- 一个 dt 同多个 dd 元素，参见图 15.8.7 中的 Writers。
- 多个 dt 元素同一个 dd，如图 15.8.4 所示。（样式设置的示例如图 15.8.5 和图 15.8.6 所示。）
- 多个 dt 元素同多个 dd 元素。如果图 15.8.4 中的 bogeyman/boogeyman 有多个定义，就是这种情形。

可以使用 dfn 元素包围 dt 中的名称，指出该列表是用于定义术语的，如在术语表中，如图 15.8.4 所示。（关于 dfn，参见 4.10 节。）

可以对描述列表进行嵌套（如图 15.8.7 所示），并通过 CSS 对它们添加所需的样式（如图 15.8.8 所示）。在默认情况下，如果一个 dl 嵌套在另一个 dl 中，它会自动进行缩进，如图 15.8.9 所示（当然也可以通过 CSS 对此进行修改）。

```
...
<body>

<h1>List of Horror Movie Legends</h1>

<dl>
    <dt>Boris Karloff</dt>
    <dd>Best known for his role in <cite>Frankenstein</cite> and related horror films,
    this
    → scaremaster's real name was William Henry Pratt.</dd>

    <dt>Christopher Lee</dt>
    <dd>Lee took a bite out of audiences as Dracula in multiple Hammer horror
    classics.</dd>

    ...
</dl>

</body>
</html>
```


图 15.8.1 这是最基本的定义列表，每个名 – 值组都由一个 dt 和一个 dd 构成。每个组之间的空行仅仅是为了提高代码的可读性而添加的。这些空行不是必需的，它们不改变内容的含义，也不影响内容的呈现

```
dt {
    font-weight: bold;
}
```

图 15.8.2 你可能想为 dt 元素中的术语添加一些格式，从而让它们突出显示（参见图 15.8.3）

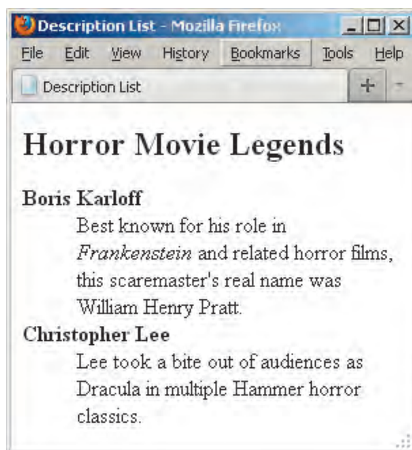


图 15.8.3 在默认情况下，名称（dt）向左对齐，而值（dd）则有缩进。由于图 15.8.2 中设置的简单样式，名称以粗体显示。如果没有设置该样式，它们将以常规文本显示

```
...
<body>

<h1>Defining words with multiple spellings</h1>

<dl>
    <dt><dfn>bogeyman</dfn>, n.</dt>
    <dt><dfn>boogeyman</dfn>, n.</dt>
    <dd>A mythical creature that lurks under
    → the beds of small children.</dd>

    <dt><dfn lang="en-gb">aluminium
    → </dfn>, n.</dt>
    <dt><dfn>aluminum</dfn>, n.</dt>
    <dd>...</dd>
</dl>

</body>
</html>
```

图 15.8.4 在这个例子中，每个名 – 值组都包含多个 dt 和一个 dd，因为这些术语有一个以上的拼写方式，而它们的定义是相同的

```
dd + dt {
    margin-top: 1em;
}
```

图 15.8.5 这会在每个名 – 值组之间添加比默认设置更多的空间



图 15.8.6　现在可以清楚地看到各组描述之间的间隔了。由于 *aluminium, n.* 所在的 dt 紧跟在上一名 – 值组中的 dd 后面，因此图 15.8.5 中的样式规则发挥了作用

```
...
<body>
<h1>Credits for <cite>Am&eacute;lie</cite></h1>

<dl>
    <dt>Director</dt>
    <dd>Jean-Pierre Jeunet</dd>

    <dt>Writers</dt><dd>Guillaume Laurant
    ➝ (story, screenplay)</dd>
    <dd>Jean-Pierre Jeunet (story)</dd>

    <dt>Cast</dt>
    <dd>
       <!-- 开始嵌套列表-->
       <dl>
          <dt>Audrey Tautou</dt> <!-- Actor/
          ➝ Actress -->
          <dd>Am&eacute;lie Poulain</dd>
          ➝ <!-- 角色 -->

          <dt>Mathieu Kassovitz</dt>
          <dd>Nino Quincampoix</dd>
          ...
       </dl>
       <!-- 结束嵌套列表 -->
    </dd>
    ...
</dl>
</body>
</html>
```

图 15.8.7　这是一个描述列表的例子，它描述了一部电影的导演、编剧和演员表，其中的演员表是由演员姓名及其角色构成的嵌套描述列表。可以根据需要对嵌套的列表设置不同的样式（如图 15.8.8 所示）

```
dt {
    font-weight: bold;
    text-transform: uppercase;
}

/* 为任何处于某dl之内的另一个dl的dt设置样式 */
dl dl dt {
    text-transform: none;
}

dd + dt {
    margin-top: 1em;
}
```

图 15.8.8　我想对主要列表中的术语和嵌套列表中的术语进行区分，因此我对 dt 元素使用了大写字母样式，再将位于嵌套 dl 中的 dt 元素重新设为常规样式（使用 text-transform: none; 声明）。不过，注意所有的术语均以粗体显示（如图 15.8.9 所示），这是因为第一条样式规则中的声明适用于所有的 dt 元素，同时并未在嵌套列表的样式中清除这一样式

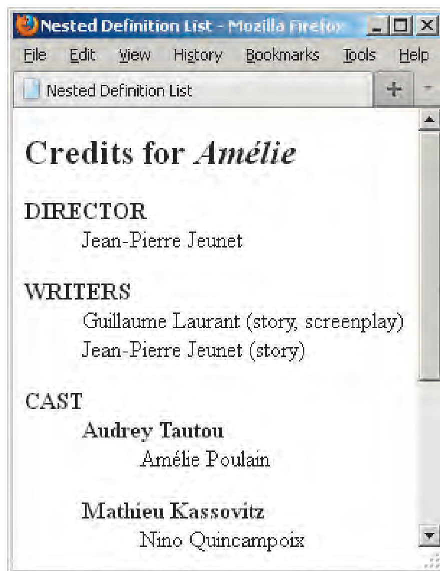


图 15.8.9　在默认情况下，当一个 dl 嵌套在另一个 dl 中时，嵌套的列表会自动进行缩进。根据图 15.8.8 中的样式，第一级 dt 元素使用大写字母，而嵌套列表中的 dt 元素则使用常规样式。所有的 dt 元素均以粗体显示

创建描述列表的步骤

(1) 输入 <dl>。

(2) 输入 <dt>。

(3) 输入需要描述或定义的单词或短语，包括任何额外的语义元素（如 dfn）。

(4) 输入 </dt> 以完成名 – 值组中的名称部分。

(5) 如果名 – 值组中有多于一个的名称或术语，可根据需要重复第 (2) 步至第 (4) 步（参见图 15.8.4）。

(6) 输入 <dd>。

(7) 输入第 (3) 步中输入的术语的描述。

(8) 输入 </dd> 以完成名 – 值组中的描述（值）部分。

(9) 如果名 – 值组中有多于一个的定义值，可根据需要重复第 (6) 步至第 (8) 步（参见图 15.8.7 中的 Writers）。

(10) 对每个由术语和描述构成的组，重复第 (2) 步至第 (9) 步。

(11) 输入 </dl> 以完成整个定义列表。

提示 对于描述（值），浏览器通常会在其术语（名称）下面新的一行对其进行缩进（参见图 15.8.3）。

提示 你应该已经在示例（图 15.8.1、图 15.8.4、图 15.8.7）中看到，不必（更确切地说，不应该）使用 p 元素对 dd 元素中的单个文本段落进行标记。不过，如果**单个**描述是由一个以上的段落构成的，就**应该**在一个 dd 元素中使用多个 p 元素对其进行标记，而不是将每个段落（不使用 p 元素）放入单独的 dd。

第 16 章 表　　单

16

本章内容

- ❑ 创建表单
- ❑ 处理表单
- ❑ 通过电子邮件发送表单数据
- ❑ 对表单元素进行组织
- ❑ 创建文本框
- ❑ 创建密码框
- ❑ 创建电子邮件框、电话框和 URL 框
- ❑ 为表单组件添加标签
- ❑ 创建单选按钮
- ❑ 创建选择框
- ❑ 创建复选框
- ❑ 创建文本区域
- ❑ 让访问者上传文件
- ❑ 创建隐藏字段
- ❑ 创建提交按钮
- ❑ 使用图像提交表单
- ❑ 禁用表单元素
- ❑ HTML5 的新特性和浏览器支持情况

到目前为止，你学到的所有 HTML 都是用于帮助你将自己的想法告诉访问者的。在本章中，你将学习如何创建表单，与访问者进行交流。

表单有两个基本组成部分：访问者在页面上可以看见并填写的字段、标签和按钮的集合；以及用于获取信息并将其转化为可以读取或计算的格式的处理脚本。

构造表单的字段和按钮很直观，同创建网页的其他部分是相似的。基本的表单字段类型包括文本框、特殊的密码框、单选按钮、复选框、下拉菜单、更大的文本区域，甚至可点击的图像。每个元素都有一个名称，作为在处理表单时用于标识数据的标签。可以使用 CSS 对表单的定位和格式进行控制，让表单变得清晰且易于使用。

使用表单通常需要用到一种服务器端的语言以接收提交的信息。这需要位于 Web 服务器的代码，用于收听表单的响应，并对响应的信息进行处理。处理的方式包括将信息存入数据库，使用电子邮件发送信息，为用户提供新的信息，等等。我推荐使用 PHP 作为入门的语言。它简单明了，且非常适合制作交互式网页。

还有很多其他的用于处理表单的服务器端语言。服务器端语言超出了本书的范围，甚至解释如何使用现有的脚本都有点儿超出界限，因此我将提供一些现成的脚本帮你起步。

16.1　创建表单

表单有三个重要的组成部分：`form` 元素，其中包含用于处理表单的脚本的 URL 和处理表单的方法（`post` 或 `get`）；表单元素，

如字段和选择框（复选框、下拉菜单、单选按钮）；以及提交按钮，用于触发向服务器上的接听脚本发送数据的动作，如图 16.1.1 所示。

```
<form method="post" action="showform.
php">
    <fieldset>
        <h2 class="account">Account</h2>
        <ul>
            <li>
                <label for="first_name">First
                → Name:</label>
                <input type="text" id="first_
                → name" name="first_name"
                → class="large" />
            </li>
            <li>
                <label for="last_name">Last
                → Name:</label>
                <input type="text" id="last_
                → name" name="last_name"
                → class="large"/>
            </li>
...
        <input type="submit"
        → class="create_profile"
        → value="Create Account">
    </fieldset>
</form>
```

图 16.1.1 每个表单都有三个部分：form 元素、访问者输入信息的实际表单元素以及用于将收集到的信息发送给服务器的提交按钮（或活动图像）

关于对表单使用 method="post" 还是 method="get" 的选择有很多要考虑的细节。通常情况下，我推荐使用 method="post"，因为使用这种方法可以向服务器发送更多的数据，同时表单中的信息不会显示在 URL 中。因此，如果需要在数据库中保存、添加和删除数据，就应选择 post。如果对表单使用 method="get"，那么表单中的数据会显示在浏览器的地址栏里，这样，用户就可以将结果存入书签。大多数搜索引擎都会在搜索表单中使用 method="get"，从而让用户可以保存搜索查询，或者将查询发给朋友。

创建表单的步骤

(1) 输入 <form method="post"。

(2) 输入 action="*script.url*">，这里的 *script.url* 是提交表单时要运行的脚本在服务器上的位置。

(3) 根据从 16.5 节开始讲解的知识，创建表单的内容（包括一个提交按钮）。

(4) 输入 </form> 以结束表单。

提示 可以在本书的配套网站下载 showform.php 脚本（www.bruceontheloose.com/htmlcss/examples/），并在第 (2) 步中使用该脚本以测试表单（在本章中都可以使用该脚本对表单进行测试）。该脚本也显示在图 16.2.1 中。

提示 为了让访问者将表单中的数据发送给你，需要包含一个提交按钮。

提示 可以使用 CSS 对表单元素进行布局（如图 16.1.2 所示）。贯穿本章的表单示例如图 16.1.3 所示。

提示 还可以使用 get 方法对通过表单收集到的信息进行处理。不过，由于 get 方法对一次性可以收集的数据的量有限制，而这个表单中有一个文件上传框，因此我推荐使用 post 方法。

```
fieldset {
     background-color: #f1f1f1;
     border: none;
     border-radius: 2px;
     margin-bottom: 12px;
     overflow: hidden;
     padding: 0 10px;
}

ul {
     background-color: #fff;
     border: 1px solid #eaeaea;
     list-style: none;
     margin: 12px;
     padding: 12px;
}

ul li {
     margin: 0.5em 0;
}

label {
     display: inline-block;
     padding: 3px 6px;
     text-align: right;
     width: 150px;
     vertical-align: top;
}

input, select, button {
     font-size: 100%;
}

.small {
     width: 75px;
}
.medium {
     width: 150px;
}
.large {
     width: 250px;
}
```

图 16.1.2　这是用于对表单进行格式化的样式表的一部分。完整的样式表见本书的配套网站 www.bruceontheloose.com/htmlcss/examples/

图 16.1.3　这是本章讨论的完整的 New Account 表单

16.2　处理表单

表单从访问者那里收集信息，脚本则对这些信息进行处理。脚本可以将信息记录到服务器上的数据库里，通过电子邮件发送信息，或者执行很多其他的功能。

在本章中我们主要关注的是 Web 表单的创建，因此我们会使用一个非常简单的 PHP 脚本，当访问者填写并提交表单时，这段脚本会将数据返回给访问者，如图 16.2.1 所示。我还将提供另一个脚本，用于将表单的内容发送到你的电子邮箱，如图 16.2.2 所示。

```
<!DOCTYPE html>
<html lang="en">
    <head>
    <meta charset="utf-8" />
       <title>Processing Form Data</title>
    <style type="text/css">
    body {
       font-size: 100%;
       font-family: Arial, sans-serif;
    }

    </style>
</head>
<body>
<p>This is a very simple PHP script that outputs the name of each bit of information (that
→ corresponds to the <code>name</code> attribute for that field) along with the value that was sent
→ with it right in the browser window.</p>
<p>In a more useful script, you might store this information in a MySQL database, or send it to your
→ email address.</p>
<table>
<tr><th>Field Name</th><th>Value(s)</th></tr>

<?php
if (empty($_POST)) {
    print "<p>No data was submitted.</p>";
} else {

foreach ($_POST as $key => $value) {
    if (get_magic_quotes_gpc()) $value=stripslashes($value);
    if ($key=='extras') {

    if (is_array($_POST['extras']) ){
       print "<tr><td><code>$key</code></td><td>";
       foreach ($_POST['extras'] as $value) {
             print "<i>$value</i><br />";
             }
             print "</td></tr>";
       } else {
       print "<tr><td><code>$key</code></td><td><i>$value</i></td></tr>\n";
       }
    } else {

    print "<tr><td><code>$key</code></td><td><i>$value</i></td></tr>\n";
    }
}
}
?>
</table>
</body>
</html>
```

图 16.2.1　这是用于处理本章表单的脚本。注意 PHP 脚本放置在 HTML 页面内（可以在本书配套网站上找到这段脚本的带注释版本，见 www.bruceontheloose.com/htmlcss/examples/）

Processing Form Data

Field Name	Value(s)
first_name	Rose
last_name	Wood
email	rose.wood@testemail.com
password	pass1234
password2	pass1234
street_address	27 Carpenter Street
city	Maplewood
state	CA
zip_code	92102
picture	rose_pic.jpg
screen_name	rose1234
web_site	http://www.rosewood.com
bio	My name is Rose Wood, and I'm excited to be here.
gender	female
email_ok_msg_from_users	on
email_ok_occasional_updates	on

图 16.2.2　图 16.2.1 中的脚本会在浏览器窗口中将每个字段的名和值通过表格呈现出来

1. 关于 PHP

PHP 是一个递归的缩写词，它表示 PHP: Hypertext Preprocessor。这是一种开源的脚本语言，它是专门为制作交互式 Web 页面而设计的。它非常简单明了 [要了解更多关于使用 PHP 的信息，我强烈推荐 Larry Ullman 的 *PHP for the Web: Visual QuickStart Guide*，*Fourth Edition*[①] 一书（Peachpit Press，2011）。] 我的脚本称不上复杂，恰好能体现 PHP 的简单性。不需要花费太多的工夫，就能让脚本做我需要的任何事情。

从基本的服务器端 Web 任务，到复杂的 Web 应用（如 WordPress 和 Drupal，它们是最流行的博客与内容管理系统），PHP 都能胜任。PHP 既是一种入门级的编程语言，也是一种专业的编程语言。随着你技能的增长，PHP 也能跟上！

除了容易学习以外，PHP 还有很多其他的特性让它成为处理HTML表单的理想语言。首先，PHP 是一种解释型（又称脚本）语言，这意味着它在执行前不需要先进行编译。写完 PHP，马上就可以执行。其次，PHP 脚本可以是独立的文本文件，但也可以直接在 HTML 页面中编写，这使得 PHP 对 Web 设计人员来说特别方便。

最后，由于 PHP 是专门为万维网设计的，因此它非常适合执行网页所需的任务，并能与 HTML 很好地协作。它有数百个内置函数可供利用。在本章中，我们只会简单地讨论 PHP 表单处理工具。PHP 的官方网站为 www.php.net。

服务器端与客户端

PHP 是一种**服务器端**（server-side）语言，这意味着它运行于为网页服务的计算机（称为**服务器**），而不是查看网页的访问者的计算机。脚本必须上传到服务器上才能发挥作用。此外，服务器必须已经安装 PHP 才能对脚本进行解释。专业网站的很多功能（如存储数据，发送电子邮件等）都需要服务器端语言。

客户端（client-side）语言（如 JavaScript）是在浏览器中运行的。它们可以在完全不与服务器交互的情况下执行很多任务。对于操纵浏览器窗口，在提交表单前检查是否所有数据都已填写，以及其他不需要服务器（或在涉及服务器之前）的任务，客户端语言都很适合。

① 本书中文版《PHP 基础教程（第 4 版）》已由人民邮电出版社出版，详细信息参见图灵社区本书页面 http://www.ituring.com.cn/book/741。——编者注

2. 安全性

向服务器发送数据的时候，尤其需要注意安全性。不要对数据作任何假设，因为即便对表单建立了安全措施，坏人也可以创建他们自己的表单，调用你的脚本并发送无数的垃圾信息。应该显式地检查数据，确保它们是符合预期的，不包含任何额外的有害内容。

3. PHP 的替代语言或框架

有很多其他的语言或框架可以替代 PHP 对表单进行处理，如 Microsoft 的 ASP.NET、Adobe 的 ColdFusion，以及 JSP（Java Server Pages）、Ruby on Rails 等。

16.3　通过电子邮件发送表单数据

如果不想为服务器端脚本操心，并且能够处理未经格式化（或未经脚本预处理）的数据，可以选择通过电子邮件发送访问者的数据，如图 16.3.1 ～图 16.3.4 所示。

```
...
<body>

<?php
//这是一个非常简单的PHP脚本……
if (empty($_POST)) {
    print "<p>No data was submitted.</
p>";
    print "</body></html>";
    exit();
}

function clear_user_input($value) {
    if (get_magic_quotes_gpc())
    → $value=stripslashes($value);
    $value= str_replace( "\n", '',
    → trim($value));
    $value= str_replace( "\r", '', $value);
    return $value;
    }

$body ="Here is the data that was
→ submitted:\n";

foreach ($_POST as $key => $value) {
```

（接右栏代码）

```
    $key = clear_user_input($key);
    $value = clear_user_input($value);
    if ($key=='extras') {

    if (is_array($_POST['extras']) ){
      $body .= "$key: ";
      $counter =1;
      foreach ($_POST['extras'] as
      → $value) {
            //除了最后一个元素，都应该添加逗号
            和空格
            if (sizeof($_POST['extras'])
            → == $counter) {
              $body .= "$value\n";
              break;}
            else {
              $body .= "$value, ";
              $counter += 1;
              }
            }
      } else {
      $body .= "$key: $value\n";
      }
    } else {

    $body .= "$key: $value\n";
    }
}

extract($_POST);
$email = clear_user_input($email);
$first_name = clear_user_input
→ ($first_name);

$from='From: '. $email . "(" . $first_
→ name . ")" . "\r\n" . 'Bcc: yourmail@
→ yourdomain.com' . "\r\n";


$subject = 'New Profile from Web Site';


mail ('yourmail@yourdomain.com',
$subject,
→ $body, $from);
?>

<p>Thanks for your signing up!</p>
</table>
</body>
</html>
```

图 16.3.1　这是用于通过电子邮件发送数据的脚本。可以在本书的配套网站找到这段脚本的带注释版本

16

图 16.3.2 除了对 action 字段进行了更新，这个表单与图 16.1.3 是一样的

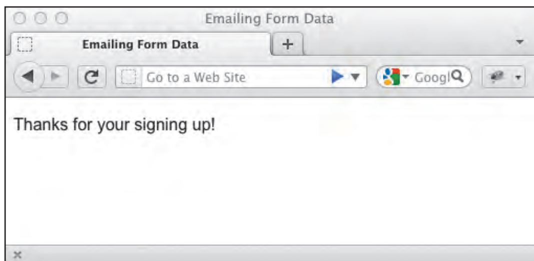


图 16.3.3 让访问者知道刚刚发生了什么总是一个好主意

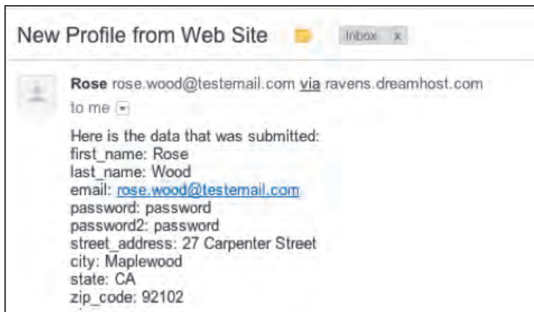


图 16.3.4 这是表单提交后收到的电子邮件

通过电子邮件发送数据的步骤

(1) 输入 `<form method="post"`。

(2) 输入 `action="emailform.php"`，这里的 *emailform.php* 是将数据发送到电子邮件地址的脚本。

(3) 输入 `>`。

(4) 根据从 16.5 节开始讲解的知识，创建表单的内容。

(5) 输入 `</form>`。

提 示 可以要求访问者输入两遍电子邮件地址，然后让脚本对这两个字段进行比较，如果不一致就返回错误。这种验证可以防止输入错误，这些错误会导致收不到表单数据。

提 示 可以在本书的配套网站找到脚本代码（www.bruceontheloose.com/htmlcss/examples/）。你可以在自己的网站上使用该脚本。

提 示 如果这段脚本在你的服务器上不起作用，这可能是因为你的服务器没有安装 PHP。请联系你的 Web 主机提供商并请求帮助（或查看他们的支持页面）。

16.4 对表单元素进行组织

如果表单上有很多信息需要填写，可以使用 fieldset 元素将相关的元素组合在一起，使表单更容易理解，如图 16.4.1 所示。表单越容易让访问者理解，访问者就越有可能正确地填写表单。可以使用 legend 元素为每个 fieldset 提供一个标题（caption），用于描述每个组的目的，如图 16.4.2 所示。

对表单元素进行组织的步骤

(1) 在 form 开始标记的下面、任何希望

包含在第一个组的表单元素的上面，输入 <fieldset>。

(2) 如果需要，输入 <legend。

(3) 如果需要，输入 align="*direction*"，这里的 *direction* 可以是 left 或 right。

(4) 输入 >。

(5) 输入标签的文本。

(6) 输入 </legend> 以完成标签。

(7) 创建属于第一组的表单元素。更多信息参见从 16.5 节开始讲解的知识。

(8) 输入 </fieldset> 结束第一组表单元素。

(9) 为每一组表单元素重复第 (1) 步至第 (8) 步。

提 示　CSS 对 legend 元素的定位能力很有限，这是在大多数浏览器中对该元素设置样式都受到限制的原因（如图 16.4.3 所示）。推荐使用具有恰当样式的 p 或 h1 ~ h6 元素重新创建标题的效果，如图 16.4.4、图 16.4.5 和图 16.4.6 所示。

提 示　使用 fieldset 元素对表单进行组织是可选的。

```
<form method="post" action="showform.php">
<fieldset>
    <legend>Account</legend> </fieldset>
<fieldset>
    <legend class="address">Address
    → </legend>
</fieldset>
<fieldset>
    <legend class="public-
    profile">Public
    → Profile</legend>
</fieldset>
```

16.4.1　我为 fieldset 元素添加了样式，并为每个 legend 元素添加了 class 属性，以帮助对每组表单元素应用样式

```
fieldset {
    background-color: #f1f1f1;
    border: none;
    border-radius: 2px;
    margin-bottom: 12px;
    overflow: hidden;
    padding: 0 10px;
}

legend {
    background-color: #dedede;
    border-bottom: 1px solid #d4d4d4;
    border-top: 1px solid #d4d4d4;
    border-radius: 5px;
    box-shadow: 3px 3px 3px #ccc;
    color: #fff;
    font-size: 1.1em;
    margin: 12px;
    padding: 0.3em 1em;
    text-shadow: #9FBEB9 1px 1px 1px;
    text-transform: uppercase;
}

legend.account { background-color: #0B5586; }
legend.address { background-color: #4494C9; }
legend.public-profile { background-color:
→ #377D87; }
legend.emails { background-color: #717F88; }
```

图 16.4.2　我为所有的 fieldset 元素添加了外边距、背景颜色和内边距，并为每个标签添加了特殊的背景颜色

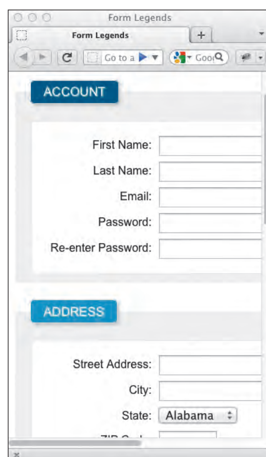


图 16.4.3　浏览器限制了对 legend 元素添加样式的能力。如果想更好地控制标签的格式，可以使用带类名的标题或 p 元素

16

```
<form method="post" action="showform.php">
    <fieldset>
        <h2 class="account">Account</h2>
```

图 16.4.4　由于大多数浏览器对 legend 元素的样式控制能力不足，因此推荐使用常规的标题元素

```
h2 {
    background-color: #dedede;
    border-bottom: 1px solid #d4d4d4;
    border-top: 1px solid #d4d4d4;
    border-radius: 5px;
    box-shadow: 3px 3px 3px #ccc;
    color: #fff;
    font-size: 1.1em;
    margin: 12px;
    padding: 0.3em 1em;
    text-shadow: #9FBEB9 1px 1px 1px;
    text-transform: uppercase;
}
```

图 16.4.5　这里对 legend h2 添加了背景、边框以及其他的一些稍后即将了解的 CSS3 特性

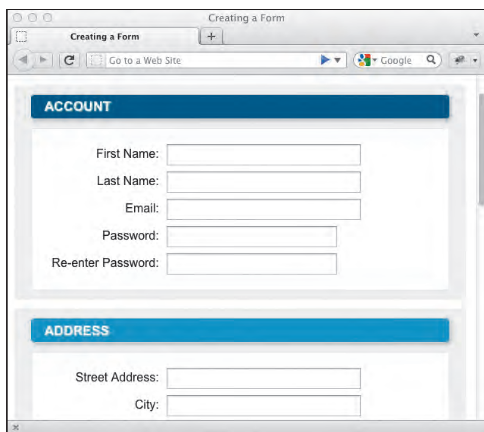


图 16.4.6　现在，标签拥有了更多的样式设置选项

16.5　创建文本框

文本框可以包含一行无格式的文本，它可以是访问者想输入的任何内容，通常用于姓名、地址等信息。

对各个表单元素进行分隔有很多方法。在这些例子中，我们使用无序列表，如图 16.5.1 和图 16.5.2 所示，不过，使用 div、p 或 br 元素对表单元素进行组织也是可行的。

```
<form method="post" action="showform.php">
<fieldset>
    <h2 class="account">Account</h2>
    <ul>
        <li>
            <label for="first_name">First
            ➝ Name:</label>
            <input type="text" id="first_
            ➝ name" name="first_name"
            ➝ class="large" required=
            ➝ "required"
            placeholder="Enter
            ➝ your first name" />
        </li>
        <li>
            <label for="last_name">Last Name:
            ➝ </label>
            <input type="text" id="last_
            ➝ name" name="last_name"
            ➝ class="large"/>
        </li>
```

图 16.5.1　必须为每个文本框设置 name 属性；只有在希望为文本框添加默认值的情况下才需要设置 value 属性

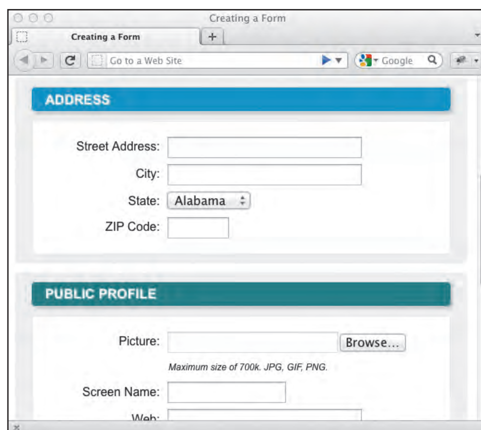


图 16.5.2　文本框可以设置为不同的大小，以容纳不同类型的字段。这个例子是通过类设置宽度的，还可以通过 HTML 元素上的 size="n" 属性设置宽度

> **新的 HTML5 表单输入属性**
>
> 表单是令大多数开发人员感到头疼的 HTML 领域之一，因为通常需要编写额外的 CSS 和 JavaScript 才能让表单正常运行。为了让表单更易于使用，HTML5 对它进行了大量的增强，其中的很多特性现在就可以使用。
>
> 这些属性包括 autofocus、required、placeholder、maxlength 和 pattern。在接下来的例子中，你将逐一学习这些特性。
>
> 不支持这些新特性的旧浏览器会忽略这些属性。很多开发人员使用 JavaScript 来填补这些浏览器在功能上的差距。
>
> 关于 HTML5 表单的现状，以及浏览器对每个特性的支持情况，参见 http://wufoo.com/html5。

创建文本框的步骤

(1) 如果需要，输入用于让访问者识别文本框的标签（例如 Name:）。

(2) 输入 <input type="text"。

(3) 输入 name="*label*"，这里的 *label* 是用于让服务器（和脚本）识别输入数据的文本。

(4) 输入 id="*label*"，这里的 *label* 是用于让对应的标签元素识别文本框的文本，后面很快就会解释这一点。它还用于让 JavaScript 为表单添加功能。很多程序员将 id 和 name 设为同一个值，尽管这并不是必需的。

(5) 如果需要，输入 value="*default*"，这里的 *default* 是这个字段中最初显示的数据，如果访问者没有输入别的内容的话，这一数据将被发送到服务器。

(6) 如果需要，输入 required="required"，表示仅在这个字段有值的情况下才能提交表单，如图 16.5.3 所示。

(7) 如果需要，输入 placeholder="*hinttext*"，这里的 *hinttext* 是这个字段中最初显示的数据，用于指导用户的输入，如图 16.5.4 所示。当 input 元素获得焦点时，这些文本将会消失，让用户输入内容。

(8) 如果需要，输入 autofocus="autofocus"，如图 16.5.5 所示。如果这是第一个拥有此属性的 input 元素，该元素在页面加载时会默认获得焦点。

(9) 如果需要，通过输入 size="*n*" 定义文本框的大小，这里的 *n* 是需要设置的文本框宽度，以字符为单位。也可以使用 CSS 设置输入框的宽度。

(10) 如果需要，输入 maxlength="*n*"，这里的 *n* 是该文本框允许输入的最大字符数。

(11) 最后，输入 />，结束文本框。

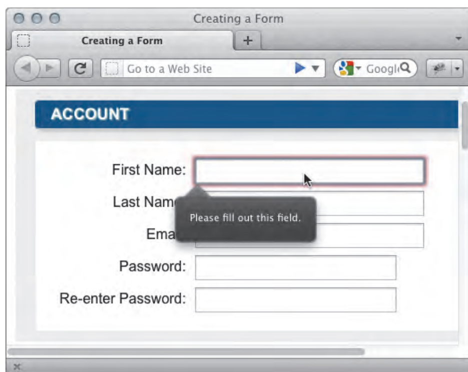


图 16.5.3 可以将文本框指定为必须填写内容才能提交表单。可以通过添加 required 或 required="required" 属性指定文本框为必填字段。这是 HTML5 的新特性，旧的浏览器将忽略这一属性。即便添加了这一属性，也应该在服务器端对表单进行验证，同时，可以使用 JavaScript 在浏览器中对字段进行检查

> **提 示** 即便访问者跳过了某个字段（而你没有使用 value 属性设置默认文本），name 属性仍将被发送到服务器（使用未定义的空值）。

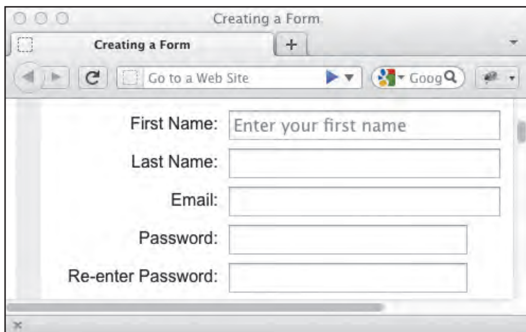


图 16.5.4　占位符是为用户填写表单提供提示和额外指导的好方法。placeholder 属性中的文本会以浅灰色出现在文本框中。当用户开始在字段中输入文本时，浅灰色文本就会消失。如果用户在没有输入任何信息的情况下将焦点移开，浅灰色文本又会再次出现。这是 HTML5 的另一项新特性，旧浏览器会忽略它

```
<input type="text" id="first_name" name=
→"first_name" class="large" required=
→"required" placeholder="Enter your first
→name" autofocus="autofocus" />
```

图 16.5.5　让某个字段在页面加载时自动获得焦点是很有益处的，这样，用户就可以立即开始输入。使用 autofocus="autofocus" 属性可以让浏览器将光标放置在第一个拥有此属性的表单元素中

提 示　文本框大小的默认值为 20。不过，访问者可以输入更多的字符，直到达到 maxlength 属性规定的限制。如果需要更大的可以容纳好几行文本的字段，最好使用文本区域。

提 示　不要将 placeholder 属性同 value 属性弄混。它们都会让文本框默认出现一些文本，但 placeholder 文本会自动消失，且不会被发送到服务器，而 value 在输入框获得焦点时不会消失，且这些内容会被发送到服务器。

16.6　创建密码框

密码框与文本框的唯一区别是，密码框中输入的文本会使用圆点或星号进行隐藏，如图 16.6.1 和图 16.6.2 所示。不过，这些信息在发往服务器时并未加密。

```
<li>
    <label for="password">Password:</label>
    <input type="password" id="password"
    → name="password" />
</li>
<li>
    <label for="password">Re-enter Password:
    → </label>
    <input type="password" id="password2"
    → name="password2" />
</li>/>
```

图 16.6.1　编译数据时，password 属性会识别密码。id 属性用于添加样式，并提供对标签的引用。type 属性必须设为 password，但 id 和 name 可以是任何不包含空格的值

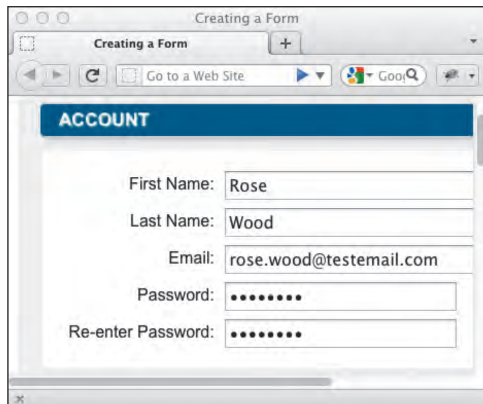


图 16.6.2　当访问者在表单中输入密码时，密码用圆点或星号隐藏起来了

创建密码框的步骤

(1) 输入用于让访问者识别密码框的标签，例如 <label for="*label*">Password</label>，这里的 *label* 应与第 (4) 步中的标签相同。关于标签，参见 16.8 节。

(2) 输入 <input type="password"。

(3) 输入 name="*label*"，这里的 *label* 是用于让服务器（和脚本）识别输入数据的文本。

(4) 输入 id="*label*"，这里的 *label* 是用于让对应的标签和 JavaScript 识别输入字段的文本。

(5) 如果需要，输入 required="required"，表示仅在这个字段有值的情况下才能提交表单。

(6) 通过输入 size="*n*" 定义密码框的大小，这里的 *n* 是需要设置的密码框宽度，以字符为单位。也可以使用 CSS 设置输入框的宽度。

(7) 输入 maxlength="*n*"，这里的 *n* 是该密码框允许输入的最大字符数。

(8) 最后，输入 />，结束密码框。

提示 即便密码框中没有输入任何内容，*name* 属性仍将被发送到服务器（使用未定义的 *value*）。

提示 密码框提供的唯一保护措施就是防止其他人看到用户输入的密码。如果要真正地保护密码，可以使用安全服务器（https://）。

16.7 创建电子邮件框、电话框和 URL 框

电子邮件、电话和 URL 这几种输入类型是 HTML5 中新增的。它们看起来同文本框很相似，但却有一些小而有用的特性，用于帮助验证和输入内容，如图 16.7.1、图 16.7.2 和图 16.7.3 所示。浏览器对这些新字段的支持程度正在增加，而旧的浏览器会将它们当做文本框进行处理，如图 16.7.4 所示。

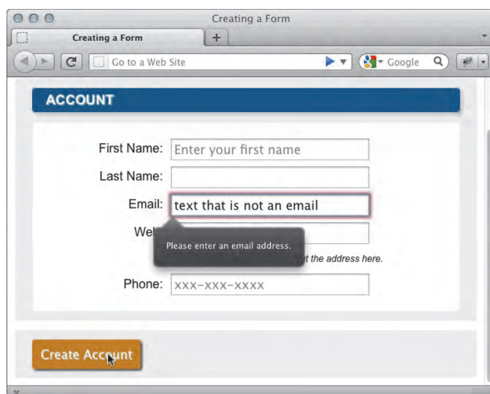


图 16.7.1 当访问者在电子邮件字段中输入文本时，浏览器会对输入的文本进行检查，确保它是有效的电子邮件地址格式。如果没有添加 required 属性，那么空的字段也可以通过验证

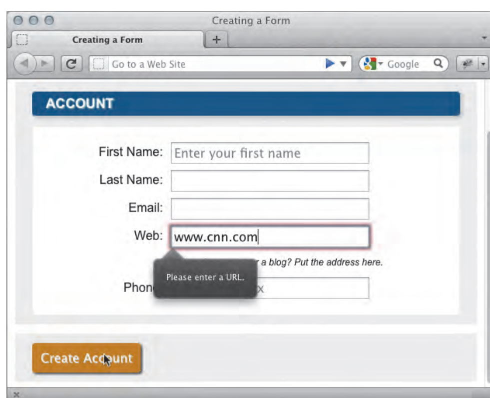


图 16.7.2 当访问者在 Web 字段中输入文本时，浏览器会对输入的文本进行检查，确保它是有效的 URL 格式。注意，www.cnn.com 并不是有效的 URL，因为 URL 必须以 http:// 或 https:// 开头。这里最好使用占位符提示访问者

创建电子邮件框、电话框和 URL 框的步骤

(1) 如果需要，输入用于让访问者识别框的标签，例如 <label for="*idlabel*">Email</label>，这里的 *idlabel* 应与第 (4) 步中的 *idlabel* 相同。你将在下一节学习标签。

(2) 对于电子邮件框，输入 <input type=

"*email*"；对于 URL 框，输入 <input type="*url*"；对于电话框，输入 <input type="*tel*"。

(3) 输入 name="*label*"，这里的 *label* 是用于让服务器（和脚本）识别输入数据的文本。

(4) 输入 id="*idlabel*"，这里的 *idlabel* 是用于让对应的标签和 JavaScript 识别输入字段的文本。

(5) 如果需要，输入 required="required"，表示仅在这个字段有值的情况下才能提交表单。

(6) 如果需要，输入 pattern="*regex*"，这里的 *regex* 是将输入的内容限定为特定格式的正则表达式。

(7) 如果需要，通过输入 size="*n*" 定义框的大小，这里的 *n* 是需要设置的框的宽度，以字符为单位。

(8) 如果需要，输入 maxlength="*n*"，这里的 *n* 是该框允许输入的最大字符数。

(9) 最后，输入 />，结束输入框。

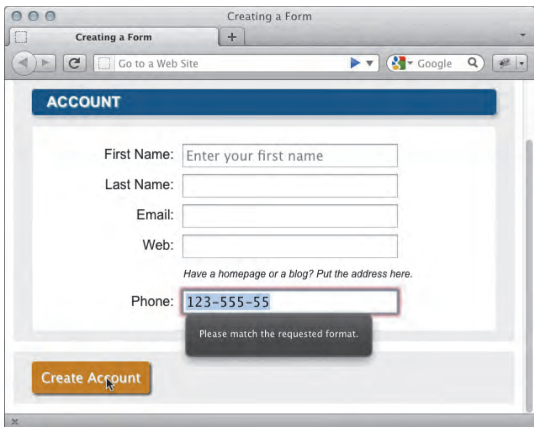


图 16.7.3　当访问者在电话字段中输入文本时，浏览器会对输入的文本进行检查，确保它符合 XXX-XXX-XXXX 的格式。同时，这种字段对于使用 iOS 上的 Safari 的用户来说也很方便，因为它会启动数字键盘而非常见的标准键盘

```
<li>
    <label for="email">Email:</label>
    <input type="email" id="email"
    ➝ name="email" class="large" />
</li>
<li>
    <label for="web_site">Web:</label>
    <input type="url" id="web_site"
    ➝ name="web_site" class="large" />
    <p class="instructions">Have a homepage or
    a blog? Put the address here.</p>
</li>
<li>
    <label for="phone">Phone:</label>
    <input type="tel" id="phone"
    ➝ name="phone" placeholder=
    ➝ "xxx-xxx-xxxx" class="large"
    ➝ pattern="\d{3}-\d{3}-\d{4}" />
</li>
```

图 16.7.4　type 属性可以识别电子邮件框、URL 框和电话框。pattern 属性用于定制的验证规则。它使用正则表达式对用户在框中输入的内容进行限制。不必对那些不同寻常的正则表达式语法感到担心，可以在 http://html5pattern.com 找到一些常用的正则表达式

提 示　正则表达式超出了本书的范围，不过，网上有很多查找模式[1]的资源。要确保向用户清楚地说明他们需要遵循的模式。如果你不仔细，访问者就有可能放弃提交表单，并且一去不复返。不要让这种事情发生！

16.8　为表单组件添加标签

HTML 提供了对标签进行标记的方法，这样就可以将它们与相关联的元素正式地连接起来，还可以用于编写脚本和其他目的。目前，你已经在一些例子中看到了带 for 属性的 label 元素，用于对每个表单元素提供解释性信息。

① 模式指的是用于描述一系列用于匹配规则的字符串（表达式）。——译者注

例如，在访问者应该输入其姓名中的名的文本字段之前，可能有“First Name:”的字样，如图 16.8.1 所示。

```
<fieldset>
    <h2 class="account">Account</h2>
    <ul>
        <li>
            <label for="first_name">First
            ➝ Name:</label>
            <input type="text" id="first_
            ➝ name" name="first_name"
            ➝ class="large" />
        </li>
        <li>
            <label for="last_name">Last Name:
            ➝ </label>
            <input type="text" id="last_
            ➝ name" name="last_name"
            ➝ class="large"/>
        </li>
    </ul>
</fieldset>
```

图 16.8.1　以正式的方式对标签进行标记提供了一种在样式表中识别它们的简单方法。如果在 label 中使用 for 属性，该属性的值应该与对应表单元素的 id 属性值相同

有时，placeholder 被错误地用做了 label。应该只在需要提供提示信息的情况下使用 placeholder。

为表单组件添加正式标签的步骤

(1) 输入 <label。

(2) 如果需要，输入 for="*idname*">，这里的 *idname* 是对应表单元素的 id 属性值。

(3) 输入标签的内容。

(4) 输入 </label>。

提 示　如果使用 for 属性，就必须同时在相关联的 form 元素的开始标记中添加 id 属性，才能使 for 属性生效（否则，文档无法通过验证）。

提 示　如果省略 for 属性，那么对于需要添加标签的元素，id 属性并不是必需的。在这种情况下，标签和该元素之间是通过邻近原则（如位于同一个 li 元素）联系在一起的。

提 示　另一种添加标签的技术是使用 title 属性。更多信息参见 3.16 节。另外，placeholder 属性无法完全替代标签。

提 示　可以使用 CSS 对标签添加格式，如图 16.8.2 和图 16.8.3 所示。

```
label {
    display: inline-block;
    padding: 3px 6px;
    text-align: right;
    width: 150px;
    vertical-align: top;
}
```

图 16.8.2　为字段的标签添加样式可以让表单变得更为美观和用户友好

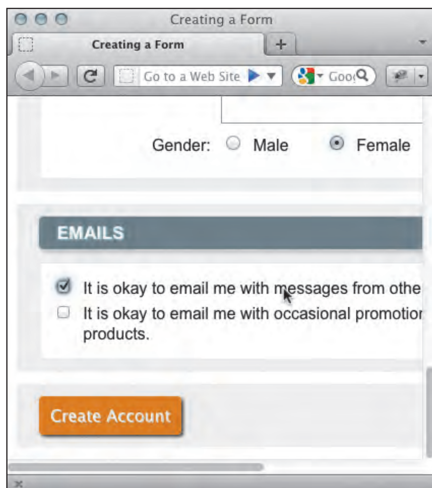


图 16.8.3　对于单选按钮和复选框的标签，用户不仅可以通过点击表单元素改变其状态，还可以通过点击标签改变表单元素的状态

16.9 创建单选按钮

还记得老式汽车收音机上那些大大的黑色塑料按钮吗？按下其中的一个可以收听 WFCR；按下另一个可以收听 WRNX。不过，不可以同时按下两个按钮。单选按钮也遵循同样的工作方式（只是它们的用途不是听收音机），如图 16.9.1、图 16.9.2 和图 16.9.3 所示。

创建单选按钮的步骤

(1) 如果需要，输入单选按钮的介绍文本。例如，可以使用“选择下列选项中的一个”。

(2) 输入 `<input type="radio"`。

(3) 输入 name="*radioset*"，这里的 *radioset* 用于识别发送至脚本的数据，同时用于将多个单选按钮联系在一起，确保同一组中最多只有一个被选中。

(4) 输入 id="*id*"，这里的 *id* 用于让对应的标签识别该单选按钮。同一组单选按钮的 name 值都是相同的，但同一页面中每个元素的 id 必须是唯一的。

(5) 输入 value="*data*"，这里的 *data* 是该单选按钮被选中（无论是被默认选中还是被访问者选中）时要发送给服务器的文本（如图 16.9.3 所示）。

(6) 如果需要，输入 checked="checked" 让该单选按钮在页面打开时默认处于激活状态。在一组单选按钮中，只能对一个按钮添加这一属性。（在 HTML 中，="checked" 是可选的。）

(7) 输入 />。

(8) 输入 <label for="*id*">radio label</label>，其中，*id* 与单选按钮中的 id 值相同，*radio label* 则用于让访问者识别该单选按钮。*radio label* 的值通常与 value 的值相同，但这并不是必需的。

(9) 对同一组内的所有单选按钮，重复第 (2) 步至第 (8) 步。

```
<fieldset class="radios">
<ul>
    <li>
       <input type="radio" id="gender_male"
       ➝ name="gender" value="male" />
       <label for="gender_male">Male</label>
    </li>
    <li>
       <input type="radio" id="gender_
       ➝ female" name="gender" value=
       ➝ "female" />
       <label for="gender_female">Female
       ➝ </label>
    </li>
</ul>
</fieldset>
```

图 16.9.1　对于单选按钮，name 属性有双重功能：将同一组的单选按钮联系在一起，并在将值发送至脚本时对其进行识别。value 是很重要的，因为对于单选按钮访问者无法输入值

```
.radios {
    background: none;
    display: inline;
    margin: 0;
    padding: 0;
}

.radios ul {
    border: none;
    display: inline-block;
    list-style: none;
    margin: 0;
    padding: 0;
}

.radios li {
    margin: 0;
    display: inline-block;
}

.radios label {
    margin-right: 25px;
    width: auto;
}

.radios input {
    margin-top: 3px;
}
```

图 16.9.2　这段 CSS 可以让无序列表中的项目水平排列。标签拥有 25 像素的右侧外边距，从而对不同的单选按钮（及标签）进行分隔（如图 16.9.3 所示）

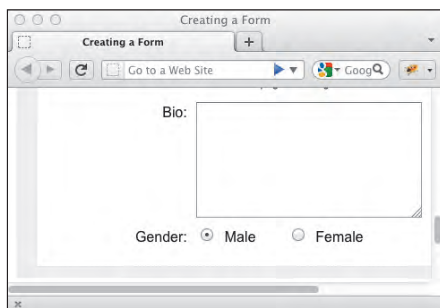


图 16.9.3 单选按钮本身是用 input 元素创建的；标签（Male 和 Female）是用 label 元素创建的。点击标签会选中对应的单选按钮

提 示 如果不设置 *value* 属性，就会向脚本发送 on。这并不是特别有用，因为你无法判断哪个按钮是被选中的。

16.10 创建选择框

选择框非常适合向访问者提供一组选项，从而允许他们从中选取。它们通常呈现为下拉菜单的样式。如果允许用户选择多个选项，选择框就会呈现为一个带滚动条的项目框，如图 16.10.1、图 16.10.2 和图 16.10.3 所示。

1. 创建选择框的步骤

(1) 如果需要，输入描述菜单的文本。

(2) 输入 <select。

(3) 输入 name="*label*"，这里的 *label* 用于在收集的数据发送至服务器时对数据进行识别。

(4) 输入 id="*idlabel*"，这里的 *idlabel* 是用于让对应的标签和 JavaScript 识别输入字段的文本。

(5) 如果需要，输入 size="*n*"，这里的 *n* 代表选择框的高度（以行为单位）。

(6) 如果需要，输入 multiple="multiple"，从而允许访问者选择一个以上的菜单选项（选择的时候需按住 Control 键或 Command 键）。

(7) 输入 >。

(8) 输入 <option。

(9) 如果需要，输入 selected="selected"，指定该选项被默认选中。

(10) 输入 value="*label*"，这里的 *label* 用于在该选项被选中时对将要发送至服务器的数据进行识别。

(11) 如果需要，输入 label="*menu option*"，这里的 *menu option* 是应该出现在菜单里的选项文字。

(12) 输入 >。

(13) 输入希望出现在菜单中的选项名称。

(14) 输入 </option>。

(15) 对每个选项重复第 (8) 步至第 (14) 步。

(16) 输入 </select>。

如果有一个选项很多的特别大的菜单，可能需要对这些选项进行分组，分别放入不同的类别。

2. 对选择框选项进行分组

(1) 根据“创建选择框的步骤”，创建所需的选择框。

(2) 在希望放在同一子菜单中的第一组选项中的第一个 option 元素之前，输入 <optgroup。

(3) 输入 label="*submenutitle*">，这里的 *submenutitle* 是子菜单的标题（如图 16.10.4 和图 16.10.5 所示）。

(4) 在该组的最后一个 option 元素之后，输入 </optgroup>。

(5) 对每个子菜单重复第 (2) 步至第 (4) 步。

```
<label for="state">State:</label>
<select id="state" name="state">
    <option value="AL">Alabama</option>
    <option value="AK">Alaska</option>
    ...
</select>
```

图 16.10.1 选择框由两种 HTML 元素构成：select 和 option。通常，在 select 元素里设置 name 属性，在每个 option 元素里设置 value 属性

16

```
select {
    font-size: 100%;
}
```

图 16.10.2　我们再次使用 CSS 对字体大小进行调整。可以使用 CSS 对 width、color 和其他的属性进行调整，不过，不同的浏览器呈现下拉菜单列表的方式略有差异

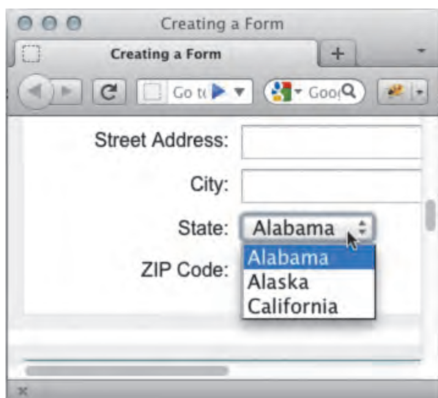


图 16.10.3　除非设置了 size 属性，否则访问者就必须选择菜单中的某个选项。默认的选择是菜单中的第一个选项，或者是在 HTML 中指定了 selected 的选项

```
<label for="referral">Where did you find out
→ about us?</label>
<select id="referral" name="referral">
<optgroup label="On-line">
    <option value="social_network">Social
    → Network</option>
    <option value="search_engine">Search
    → Engine</option>
</optgroup>
<optgroup label="Off-line">
    <option value="postcard">Postcard
    → </option>
    <option value="word_of_mouth">Word of
    → Mouth</option>
</optgroup>
</select>
```

图 16.10.4　每个子菜单都有一个标题（在 optgroup 开始标记的 label 属性中指定）和一系列选项（使用 option 元素和常规文本定义）

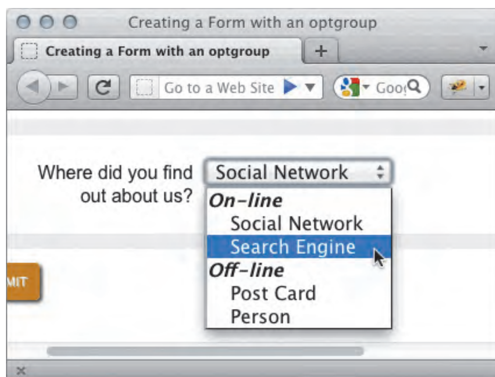


图 16.10.5　浏览器通常不会创建真正的子菜单，而是对菜单项目进行分组显示

提 示　如果添加了 size 属性，那么选择框看起来会更像一个列表，且没有自动选中的选项（除非设置了 selected）。

提 示　如果 size 大于选项的数量，访问者就可以通过点击空白区域让所有的选项处于未选中状态。

16.11　创建复选框

在一组单选按钮中，只允许选择一个答案；但在一组复选按钮中，访问者可以选择任意数量的答案。同单选按钮一样，复选框也与 name 属性的值联系在一起。

创建复选框的步骤

(1) 如果需要，输入复选框的介绍文本。例如，可以使用“选择一个或多个选项”。

(2) 输入 <input type="checkbox"。

(3) 输入 name="*boxset*"，这里的 *boxset* 用于识别发送至脚本的数据，同时用于将多个复选框联系在一起。

(4) 输入 value="*data*"，这里的 *data* 是该复选框被选中（无论是被访问者选中还是被

建站者选中，如图 16.11.1 所示）时要发送给服务器的文本。

(5) 输入 checked="checked" 让该复选框在页面打开时默认处于选中状态，建站者或访问者可能会勾选默认的选项。（在 HTML 中，="checked" 是可选的。）

(6) 输入 /> 以结束复选框。

(7) 输入 <label for="*id*"> *checkbox label* </label>，其中，*id* 与复选框元素中的 *id* 值相同，*checkbox label* 则用于让访问者识别该复选框。*checkbox label* 的值通常与 value 的值相同，但这并不是必需的。

(8) 对同一组内的所有复选框，重复第 (2) 步至第 (7) 步。

```
<ul class="checkboxes">
<li>
        <input type="checkbox" id="email_
        → ok_msg_from_users" name="email_
        → signup[]" value="user_emails"
        />
        <label for="email_ok_msg_from_users">
        → It is okay to email me with
        → messages from other users.</label>
    </li>
    <li>
        <input type="checkbox"
        → id="email_ok_occasional_
        → updates" name="email_signup[]"
        → value="occasional_updates" />
        <label for="email_ok_occasional_
        → updates">It is okay to email me
        → with occasional promotions about
        → our other products.</label>
    </li>
</ul>
```

图 16.11.1　注意，标签文本（未突出显示）不需要与 value 属性一致。这是因为标签文本用于在浏览器中向访问者标识复选框，而 value 则用于向脚本标识数据。空的方括号是为 PHP 脚本准备的（参见提示）

为复选框的标签设置样式，如图 16.11.2 和图 16.11.3 所示。

```
.checkboxes label {
    text-align: left;
    width: 475px;
}
```

图 16.11.2　对于复选框，通常需要为标签设置不同的样式，因为这些标签位于输入元素之后

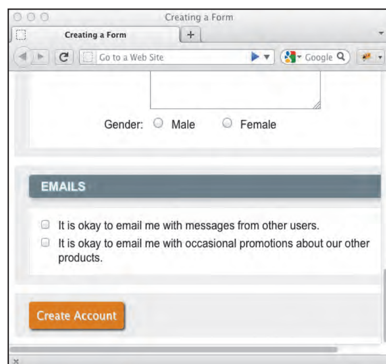


图 16.11.3　访问者可以根据需要选择任意数量的框，每个框对应的值及复选框组的名称都会被发送到脚本

提示　如果使用 PHP，在图 16.11.1 中使用 name="*boxset*[ ]"（这里的 *boxset* 用于标识发送给脚本的数据）就会自动地创建一个包含复选框值的数组（名为 $_POST['boxset']）。

16.12　创建文本区域

如果希望给访问者填写问题或评论的空间，可以使用文本区域。它们可以根据需要进行扩展，如图 16.12.1、图 16.12.2 和图 16.12.3 所示。

```
<label for="bio">Bio:</label>
<textarea id="bio" name="bio" rows="8"
→ cols="50" class="large"></textarea>
```

图 16.12.1　textarea 元素中没有 value 属性。默认值是通过在开始标记和结束标记之间添加文本进行设置的

```
textarea { font: inherit; width: 250px; }
```

图 16.12.2　在默认情况下，对于 textarea，font 属性不会继承，因此必须显式地设置该属性

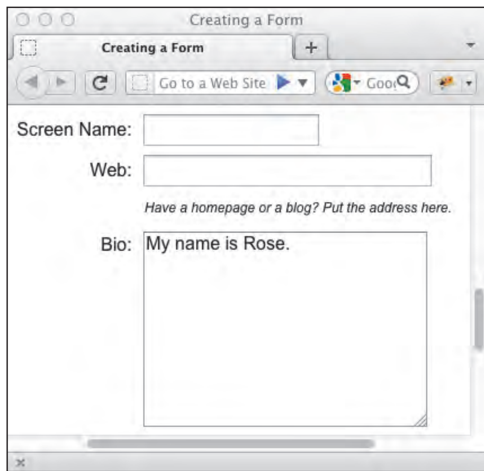


图 16.12.3　访问者可以在框中输入很多行文本

创建文本区域的步骤

(1) 如果需要，输入用于标识文本区域的解释性文本。

(2) 输入 <textarea。

(3) 输入 name="*label*"，这里的 *label* 是用于让服务器（和脚本）识别输入数据的文本。

(4) 如果需要，输入 maxlength="*n*"，这里的 *n* 是可以输入的最大字符数。对于 textarea 来说，这个属性是 HTML5 新增的，因此它在不同浏览器中的表现并不一致，参见 http://wufoo.com/html5/attributes/03-maxlength. html。

(5) 输入 rows="*n*"，这里的 *n* 是文本区域的高度（以行为单位）。

(6) 输入 cols="*n*"，这里的 *n* 是文本区域的宽度（以字符为单位）。

(7) 输入 >。

(8) 输入文本区域的默认文本（如果有的话）。

(9) 输入 </textarea> 以完成文本区域。

提 示　对于文本区域，不需要使用 value 属性。替代 value 属性的是出现在 textarea 开始标记和结束标记之间的文本。

提 示　访问者在一个文本区域中最多可以输入 327 00 个字符。必要的时候会出现滚动条。

提 示　更好的设置 textarea 的高度和宽度的方式是使用 CSS。

16.13　让访问者上传文件

有时需要让网站的用户向服务器上传文件（如照片、简历等）。

让访问者上传文件的步骤

(1) 输入 <form method="post" enctype ="multipart/form-data"。enctype 属性可以确保文件采用正确的格式上传，如图 16.13.1 所示。

(2) 接下来，输入 action="*upload.url*">，其中的 *upload.url* 是处理上传文件的脚本的 URL。为此可能需要一段专门的脚本。

(3) 为文件上传区域输入标签，如 <label for="picture">Picture:</label>。

(4) 输入 <input type="file"，创建一个文件上传框和一个 Browse（浏览）按钮，如图 16.13.2 所示。

(5) 输入 name="*title*"，这里的 *title* 用于识别将要上传的文件。

(6) 输入 id="*label*"，这里的 *label* 用于让对应的标签识别这个表单元素，它在此页

面上必须是唯一的。

(7) 如果需要，输入 size="*n*"，这里的 *n* 是访问者可以输入路径和文件名的字段的宽度。也可以使用 CSS 设置宽度。

(8) 输入 />。

(9) 跟平常一样结束表单，包括创建提交按钮，输入 </form> 结束标记。

```
<form method="post" action="showform.php"
→ enctype="multipart/form-data">
...
<label for="picture">Picture:</label>
<input type="file" id="picture" name=
→ "picture" />
<p class="instructions">Maximum size of 700k.
→ JPG, GIF, PNG.</p>
...
</form>
```

图 16.13.1 要让访问者能够上传文件，必须正确地设置 enctype 属性，创建 input type="file" 元素

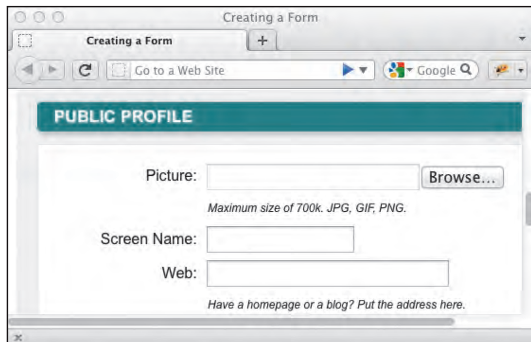


图 16.13.2 文件上传区域为用户提供了从其系统中选择文件的方式

提 示 对于需要允许上传的表单，不能使用 get 方法。

提 示 服务器在接收文件之前，需要进行正确地配置以存储文件。

16.14 创建隐藏字段

隐藏字段可以用于存储表单中的数据，但它不会显示给访问者。可以认为它们是不可见的文本框。它们通常用于存储先前的表单收集的信息，以便将这些信息同当前表单的数据一起交给脚本进行处理，如图 16.14.1 所示。

```
<form method="post" action="whatever.php">
<input type="hidden" name="name" value=
→ "<?= $name ?>" />
<input type="submit" value="submit data" />
```

图 16.14.1 创建隐藏字段时，可以使用脚本中的变量将字段的值设置为访问者原来输入的值

创建隐藏字段的步骤

(1) 输入 <input type="hidden"。

(2) 输入 name="*label*"，这里的 *label* 是对要存储的信息的简短描述。

(3) 输入 value="*data*"，这里的 *data* 是要存储的信息本身。它通常是表单处理脚本中的一个变量（参见图 16.14.1）。

(4) 输入 />。

什么时候使用隐藏字段

假设你有一个表单，希望让访问者在提交表单之前有机会检查他们输入的内容。处理表单的脚本可以向访问者显示提交的数据，同时创建一个表单，其中有包含同样数据的隐藏字段。如果访问者希望编辑数据，他们只需后退就可以了。如果他们想提交表单，由于隐藏字段已经将数据填好了，因此他们就不需要再次输入数据了。

提 示 隐藏字段出现在表单标记中的位置并不重要，因为它们在浏览器中是不可见的。只要它们位于表单的开始标记和结束标记之间即可。

提 示　如果要创建一个在访问者点击提交按钮时与其他数据一起提交的元素，同时希望访问者可以看到它，那么可以创建一个常规的表单元素并使用 readonly（只读）属性。

16.15　创建提交按钮

访问者输入的信息如果不发送到服务器，就没什么用。应该总是为表单创建提交按钮，让访问者可以将信息交给你，如图 16.15.1、图 16.15.2 和图 16.15.3 所示。（还可以使用图像提交表单数据，参见“创建带图像的提交按钮”。）

```
<input type="submit" class="create_profile"
→ value="Create Account">
```

图 16.15.1　如果不填写 name 属性，则提交按钮的名 – 值对就不会传给脚本。由于通常不需要这一信息，因此上面的做法是有益的

```
.create_profile {
     background-color: #DA820A;
     border: none;
     border-radius: 4px;
     box-shadow: 2px 2px 2px #333;
     cursor: pointer;
     color: #fff;
     margin: 12px;
     padding: 8px;
     text-shadow: 1px 1px 0px #CCC;
}
```

图 16.15.2　通过使用类，我为提交按钮应用了背景、字体格式和一些 CSS3 特性

1. 创建提交按钮的步骤

(1) 输入 <input type="submit"。

(2) 如果需要，输入 value="*submit message*"，这里的 *submit message* 是将要出现在按钮上的文本。

(3) 最后输入 />。

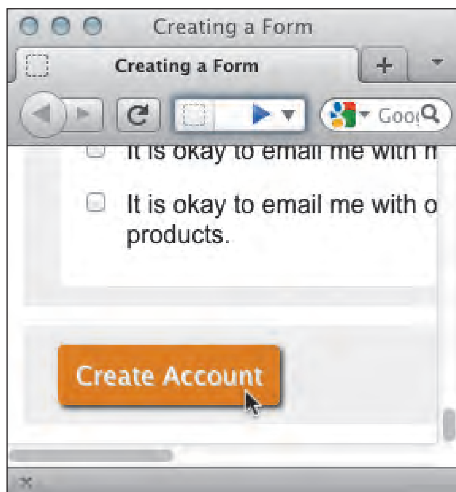


图 16.15.3　提交按钮可以激活表单收集的数据。可以通过 value 属性定制按钮的内容（对访问者来说，Create Account（创建账户）比默认文本 Submit Query（提交查询）更为明确）

2. 创建带图像的提交按钮

(1) 输入 <button type="submit">。

(2) 输入将要出现在按钮中图像左侧的文本（如果有的话）。

(3) 输入 <img src="*image.url*"，其中的 *image.url* 是将要出现在按钮上的图像的名称。

(4) 输入 alt="*alternate text*"，这里的 *alternate text* 是当图像无法显示时需要出现的文本。

(5) 如果需要，添加其他的图像属性。

(6) 输入 /> 以结束图像。

(7) 输入将要出现在按钮中图像右侧的文本（如果有的话），如图 16.15.4 和图 16.15.5 所示。

(8) 输入 </button>。

```
<button type="submit" class="create_profile">
→ Create Account</button>
```

图 16.15.4　可以使用 button 元素创建文字旁边有图像的提交按钮

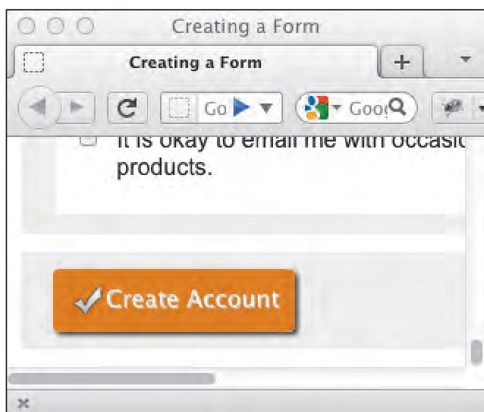


图 16.15.5 带图像的提交按钮的代码稍微复杂一些，但它可以为创建元素和为元素添加样式提供更多的控制

提示 如果省略 value 属性，那么提交按钮就会显示默认的 Submit Query。

提示 提交按钮的名–值对仅在设置了 name 属性以后才会发送给脚本。因此，如果忽略 name 属性，就不必处理这一额外的、通常没有意义的提交数据。

提示 如果有多个提交按钮，可以为每个按钮设置 name 属性和 value 属性，从而让脚本知道用户按下的是哪个按钮。

提示 还可以使用 button 元素创建不带图像的提交按钮。

提示 使用 HTML5 的 button 元素可以创建更漂亮的提交按钮，因为它可以包含其他的 HTML 元素，而不仅仅是简单的文本值。注意，不同的浏览器呈现它们的方式并不一致，因此需要进行大量的测试和 CSS 工作才能让按钮在不同的浏览器中呈现一致的显示。

16.16 使用图像提交表单

可以仅使用图像作为提交表单的输入元素。有时，设计人员创建的按钮超出了 CSS3 的能力，哪怕使用花哨的渐变、阴影和圆角也无法实现，如图 16.16.1 和图 16.16.2 所示。

```
<input type="image" alt="Create Account"
→ src="blue-submit-button.png" />
```

图 16.16.1 如果使用图像，就不需要再使用提交按钮

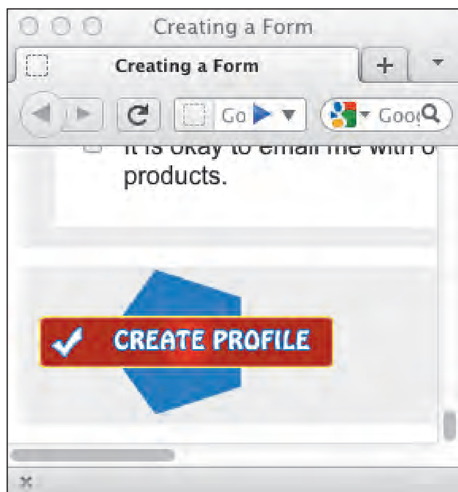


图 16.16.2 当 CSS 无法满足需求的时候，可以使用图像提交表单

使用图像提交表单的步骤

(1) 创建 PNG、GIF 或 JPEG 图像。

(2) 输入 `<input type="image"`。

(3) 输入 `src="`*`image.url`*`"`，这里的 *`image.url`* 是图像在服务器上的位置。

(4) 输入 `alt="`*`description`*`"`，这里的 *`description`* 是图像无法显示时将要出现的内容。

(5) 最后，输入 `/>` 结束用于提交表单的活动图像的定义。

16.17　禁用表单元素

在某些情况下，你可能不想让访问者使用表单中的某些部分。例如，你可能希望在所有必填字段完成之前禁用提交按钮，如图 16.17.1、图 16.17.2 和图 16.17.3 所示。

```
<li>
    <input type="radio" name="how" value=
    ➝ "facebook" id="facebook" onclick=
    ➝ "document.getElementById('other_
    ➝ description').disabled = true;" />
    <label for="facebook">Facebook</label>
</li>
<li>
    <input type="radio" name="how" value=
    ➝ "other" id="other" onclick="document.
    ➝ getElementById('other_description').
    ➝ disabled = false;" />
    <label for="other">Other</label>
</li>
<li>
    <textarea id="other_description"
    ➝ disabled="disabled"></textarea>
</li>
```

图 16.17.1　这里，通过使用 JavaScript 和 disabled 属性，让 Other 文本区域仅在 Other 单选按钮被选中时才能访问

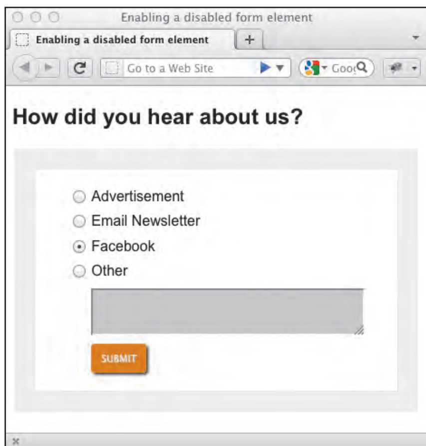


图 16.17.2　当 Other 单选按钮未选中时，文本区域是灰色的、被禁用的，用户无法选中该框并输入内容

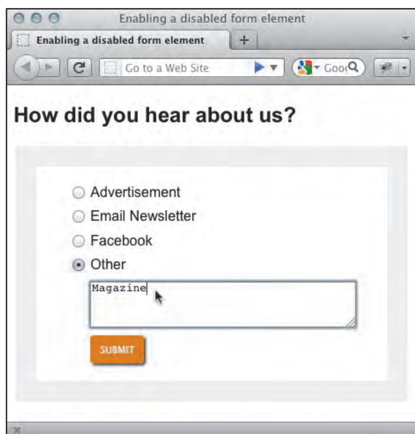


图 16.17.3　由于使用了 JavaScript，当访问者选择 Other 单选按钮以后，文本区域变成了白色，用户可以输入文本以提交到服务器

禁用表单元素的方法

在表单元素的标记中输入 disabled="disabled"（或者仅输入 disabled，这两种方式都是可行的）。

提 示　可以通过脚本修改已禁用表单元素的内容。这需要一些 JavaScript 的知识。这里使用了最简单的方法，即为单选按钮添加 onclick="document.getElementById('other_description').disabled = false;"。这样，就可以根据被选中的单选按钮对文本区域进行禁用或启用。这只是一个示例，实际上，如图 16.17.1 所示的那样将 JavaScript 混在 HTML 中是很不好的。合理的方法参见 Christian Heilmann 的文章（www.onlinetools.org/articles/unobtrusivejavascript/）。

16.18　HTML5 的新特性和浏览器支持情况

HTML5 提供了一些让创建和使用表单变得更加容易的新特性，其中的一些现在已经可以开始使用了。新的规范引入了大量新的

功能，包括新的表单元素、属性、输入类型、验证处理和样式。

还有大量的特性尚未得到广泛的支持，或者其实现还不完整。有些时候，连 W3C 的规范都没有完成。

由于旧的浏览器并不支持这些新特性，因此需要了解在如今使用这些特性时应该如何提供后备方案。

例如，`output` 就是一种已经获得广泛支持的新表单元素。它用于显示来自其他表单元素的计算值。例如，如果有一个购物车，在修改希望购买的商品的数量时，`output` 元素会在订单中显示更新的总数。该元素通常需要同 JavaScript 一起使用。

新特性的完整列表及各种浏览器对它们的支持情况见 Wufoo 的网站 The Current State of HTML5 Forms（HTML5 表单现状，http://wufoo.com/html5/）。

第 17 章

视频、音频和其他多媒体

17

本章内容

- ❑ 在网页中添加单个视频
- ❑ 为视频添加控件和自动播放
- ❑ 阻止预加载视频
- ❑ 添加具有备用超链接的视频
- ❑ 添加具有备用 Flash 的视频
- ❑ 提供可访问性
- ❑ 在网页中添加单个音频文件
- ❑ 为音频添加控件、自动播放并设置循环播放
- ❑ 预加载音频文件
- ❑ 添加具有备用超链接的音频
- ❑ 添加具有备用 Flash 的音频
- ❑ 添加同时具有备用 Flash 和超链接的音频
- ❑ 嵌入 Flash 动画
- ❑ 通过 Canvas 操作视频

万维网变得如此流行的原因之一就是可以在网页中添加图像、声音、动画和电影。虽然过去对这些文件大小的限制局限了它们的作用，但是新的技术（比如流音频和流视频，还有宽带互联网连接）已经打开了多媒体网页的大门。

有的多媒体网页可以作为音频或视频播客的基础，有的是广告或交互显示，还有的网页可以利用少量的多媒体文件让访问者获得更丰富的体验。本章将介绍如何在网页中插入多媒体，以达到上述目的以及更多的目的。

在 HTML5 出现之前，为网页添加多媒体的唯一办法就是使用第三方的插件（如 Adobe Flash Player、苹果的 QuickTime）。通过引入原生的多媒体（浏览器负责一切），HTML5 改变了上述状况。

万维网的使用人群多种多样，因此有时候要确保所有的访问者（或者说尽可能多的访问者）能看到和听到你提供的多媒体文件是件棘手的工作。你需要考虑查看或听到这些多媒体所需的文件格式。实际上，多媒体技术的开发人员还没有形成一致的标准，这让事情变得更复杂了一些。

注意，本章的目的是介绍多媒体 Web 文件，重点放在了所需的 HTML5 代码上。这里不会讲解如何创建多媒体内容，只会讲解如何让访问者可以浏览它们。

17.1 第三方插件及步入原生

如上文所述，在 HTML5 出现之前，为网页添加媒体（如音频和视频）的唯一方法就是通过第三方插件。

这些第三方插件有点像黑箱，用户必须真正安装了它们才能起作用。如果使用 Flash Player 这样的插件，用户很有可能已经安装了，因为它的市场份额很高，大多数用户系统都安装了该插件。

不过，这样做也有一些问题。在某个浏览器中嵌入 Flash 视频的代码在另一个浏览器中可能不起作用，也没有包围它的优雅方式。而且，总是存在速度方面的问题，因为浏览器会将媒体内容的播放完全交给插件。

考虑到这些问题，HTML5 规范中添加了原生的多媒体。这样做有很多好处：速度更快（任何浏览器原生的功能势必比插件要快一些），原生控件内置到浏览器，对插件的依赖极大地降低（但也并非完全不依赖，这在后面将会看到）。

对于任何一套标准，都有关于 HTML5 原生多媒体及其支持的文件格式的问题。最初，HTML5 规范指定了两种兼容 HTML5 的浏览器必须支持的媒体格式（分别对应音频和视频）。这本来是很好的，但并非所有的厂商都愿意遵循。诺基亚和苹果均未选择使用这些强制的媒体格式，因此这样的要求在规范中被移除了。这意味着你需要为你的媒体提供一种以上的格式才能让它在兼容 HTML5 的浏览器中播放。后面我们将详细地讨论这些问题。

当苹果宣布不再在其移动设备（包括 iPhone 和 iPad）上支持 Flash 时，HTML5 及原生媒体就变得更为实用了。随着这些设备的日渐普及，过去播放媒体文件对 Flash 的依赖正在迅速削减，因此迫切需要一种新的解决方案。于是，HTML5 原生多媒体进入了这一领域并展现出了强大的力量，因为苹果移动设备的浏览器是支持 HTML5 的。

闲话少叙，下面就来看看如何为网页添加原生的视频。

17.2　视频文件格式

HTML5 支持大量不同的视频文件格式（即编解码器）。

HTML5 支持三种主要的视频编解码器。以下是这三种编解码器及支持它们的浏览器。

- ❑ Ogg Theora 使用的文件扩展名为 .ogg 或 .ogv，支持它的浏览器包括 Firefox 3.5+、Chrome 5+ 和 Opera 10.5+。
- ❑ MP4（H.264）使用的文件扩展名为 .mp4 或 .m4v，支持它的浏览器包括 Safari 3+、Chrome 5-?、Internet Explorer 9+、iOS 和 Android 2+。
- ❑ WebM 使用的文件扩展名为 .webm，支持它的浏览器包括 Firefox 4+、Chrome 6+、Opera 11+、Internet Explorer 9+ 和 Android 2.3+。

提示　开发者至少需要为视频提供两种格式（MP4 和 WebM），才能确保获得所有兼容 HTML5 的浏览器的支持。这并不是很糟糕。

提示　Google 将在即将发布的 Chome 中放弃对 MP4 的支持，不过他们还没有确认这一点。

什么是编解码器

编解码器是使用压缩算法对数据的数字流进行编码和解码，使之更适合播放的计算机程序。

编解码器的目标通常是在保证音频和视频所能达到的最高质量的情况下减小文件尺寸。

当然，不同编解码器的表现是不一致的。

转换文件格式

关于如何创建视频资源的主题已经超出了本章的范围，不过如果你已经拥有一个视频资源并希望将它转换为其他的文件格式，有大量的工具可以帮你完成这一任务。下面就是两个这样的工具：

- ❑ Miro Video Converter（www.mirovideoconverter.com）
- ❑ HandBrake（http://handbrake.fr）

17.3　在网页中添加单个视频

要在 HTML5 网页中添加视频，需要使用新的 video 元素。这一过程再简单不过了，如图 17.3.1 所示。

```
<body>
    <video src="paddle-steamer.webm"></video>
</body>
```

图 17.3.1　指定单个 WebM 视频（不含控件）

在网页中添加单个视频的步骤

(1) 获取视频资源。

(2) 输入 <video src="*myVideo.ext*"></video>，这里的 *myVideo.ext* 是视频文件的位置、名称和扩展名。

就这么简单！

提 示　苹果和微软都采用了原生多媒体的概念，但又对其进行了修改。要让原生多媒体可以在 Safari（以及同样基于 WebKit 的 Chrome）中运行，用户的设备上需要安装 QuickTime；Internet Explorer 9 则需要安装 Windows Media Player，现实情况就是这样。

17.4　视频属性一览

还有哪些属性可以用在 video 元素上呢？表 17.1 列出了这些属性。

如你所见，video 元素有很多属性。它们为视频提供了很高的灵活性。

表 17.1　视频属性

属　　性	描　　述
src（源）	指定视频文件的 URL
autoplay（自动播放）	当视频可以播放时立即开始播放
controls（控件）	添加浏览器为视频设置的默认控件
muted（静音）	让视频静音（目前没有任何浏览器支持）
loop（循环）	让视频循环播放
poster（海报）	指定视频加载时要显示的图像（而不显示视频的第一帧）。接受所需图像文件的 URL
width（宽度）	视频的宽度（以像素为单位）
height（高度）	视频的高度（以像素为单位）
preload（预加载）	告诉浏览器要加载的视频内容的多少。可以是以下三个值： none 表示不加载任何视频 metadata 表示仅加载视频的元数据（如长度、尺寸等） auto 表示让浏览器决定怎样做（这是默认的设置）

17.5　为视频添加控件和自动播放

目前仅向你展示了在网页中添加视频的最简单的方法，而示例中的视频甚至不会自动开始播放，因为我们没有让它这样做。此外，如果查看这段示例代码的浏览器不支持所使用的视频文件格式，浏览器就会显示一个空的矩形（如果没有指定尺寸，则为 300 像素 × 150 像素）或海报图像（如果有的话；通过 poster 属性指定）。

17.3 节中的代码示例并未给视频添加任何控件，不过要添加控件也是相当容易的，

如图 17.5.1 所示。

```
<body>
    <video src="paddle-steamer.webm"
    ➝ controls="controls"></video>
</body>
```

图 17.5.1　添加单个 WebM 视频（这次是包含控件的）

controls 属性告诉浏览器为视频添加一套默认的控件。

布尔类型的属性

布尔类型的属性（如 controls）不需要指定值，因为它们出现在媒体元素中就足够了。

本书的示例均为这些布尔类型的属性指定了值，不过，图 17.5.1 中的 controls 也可以写做

```
<video src="paddle-steamer.webm"
➝ controls></video>.
```

每个浏览器都有自己的一套默认控件，它们之间看起来差异很大，如图 17.5.2～图 17.5.6 所示。



图 17.5.2　Firefox 中的视频控件



图 17.5.3　Safari 中的视频控件

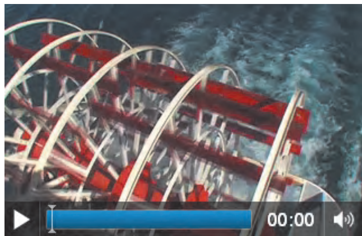


图 17.5.4　Chrome 中的视频控件

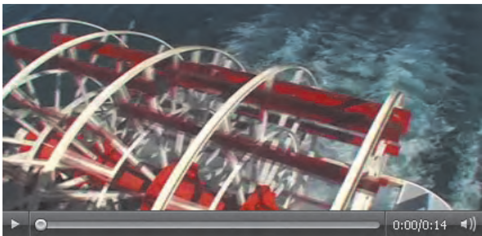


图 17.5.5　Opera 中的视频控件

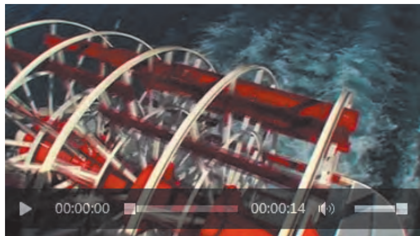


图 17.5.6　Internet Explorer 9 中的视频控件

下面的例子演示了如何使用表 17.1 中的一些视频属性，如图 17.5.7 所示。

1. 为视频添加控件的方法

输入 <video src="myVideo.ext" controls="controls"></video>。

2. 为视频添加自动播放

(1) 获取视频资源。

(2) 输入 <video src="*myVideo.ext*" autoplay="autoplay" controls="controls"></video>，这里的 *myVideo.ext* 是视频文件的位置、名称和扩展名。

```
<body>
    <video src="paddle-steamer.webm"
    → autoplay="autoplay" controls=
    → "controls"></video>
</body>
```

图 17.5.7　设置了加载时自动播放的单个 WebM 视频

17.6　为视频指定循环播放和海报图像

不仅可以将视频设为自动播放，还可以将它设为循环播放，如图 17.6.1 所示。（不过这一做法并不推荐，考虑一下用户的感受就知道原因了。）

```
<body>
    <video src="paddle-steamer.webm"
    → autoplay="autoplay" loop="loop">
    → </video>
</body>
```

图 17.6.1　设置了自动播放和循环播放的单个 WebM 视频

要实现循环播放，只需要用到 autoplay 和 loop 属性。

不过需要注意，Firefox 并不支持 loop 属性。

通常，浏览器会在视频加载时显示视频的第一帧。你可能想对此作出修改，指定你自己的图像。这可以通过海报图像实现。

1. 为视频添加自动播放和循环播放

(1) 获取视频资源。

(2) 输入 <video src="*myVideo.ext*" autoplay="autoplay" loop="loop"></video>，这里的 *myVideo.ext* 是视频文件的位置、名称和扩展名。

2. 为视频指定海报图像

(1) 获取视频资源。

(2) 输入 <video src="*myVideo.ext*" controls="controls" poster="*myPoster.jpg*"></video>，其中，*myVideo.ext* 是视频文件的位置、名称和扩展名，*myPoster.jpg* 是想用做海报图像的图像，如图 17.6.2 和图 17.6.3 所示。

```
<body>
    <video src="paddle-steamer.webm"
    → poster="paddle-steamer-poster.jpg"
    → controls="controls"></video>
</body>
```

图 17.6.2　指定了海报图像（当页面加载并显示视频时显示该图像）的单个 WebM 视频（含控件）



图 17.6.3　显示海报图像的视频。在这个例子中，图像是取自视频本身的一幅屏幕截图

17.7　阻止预加载视频

如果认为用户观看视频的可能性较低（如该视频并不是页面的主要内容），那么可以告诉浏览器不要预先加载该视频，这样能节省带宽，如图 17.7.1 和图 17.7.2 所示。

```
<body>
    <video src="paddle-steamer.webm"
    → preload="none" controls="controls">
    → </video>
</body>
```

图 17.7.1　页面完全加载时也不会加载的单个 WebM 视频。仅在用户试着播放该视频时才会加载它



图 17.7.2　将 preload 设为 none 的视频。如你所见，什么也不会显示，因为浏览器没有得到关于该视频的任何信息（连尺寸都不知道），也没有指定海报图像

告诉浏览器不预先加载视频的步骤

(1) 获取视频资源。

(2) 输入 `<video src="myVideo.ext" preload="none" controls="controls"></video>`，这里的 *myVideo.ext* 是视频文件的位置、名称和扩展名。

17.8　使用多个来源的视频

一切看起来都很棒，不过你也应该注意到了，前面所有的例子都只用了一个视频文件（同时意味着只有一种格式）。

你已经知道了，要获得所有兼容 HTML5 的浏览器的支持，至少需要提供两种格式的视频：MP4 和 WebM。

如何做到呢？这时就要用到 HTML5 的 source 元素了。

通常，source 元素用于定义一个以上的媒体元素（在这个例子中为 video）的来源。

一个 video 元素中可以包含任意数量的 source 元素，因此为我们的视频定义两种不同的格式是相当容易的，如图 17.8.1 所示。

指定两种不同的视频来源

(1) 获取视频资源（这次需要两个）。

(2) 输入 `<video controls="controls">` 开始 video 元素（含默认控件集）。

(3) 输入 `<source src="myVideo.mp4" type="video/mp4">`，这里的 *myVideo.mp4* 是 MP4 视频源文件的名称。

(4) 输入`<source src="myVideo.webm" type="video/webm">`，这里的 *myVideo.webm* 是 WebM 视频源文件的名称。

(5) 输入 `<p>Sorry, your browser doesn't support the video element</p>`，从而在浏览器不支持 HTML5 视频时显示这条消息。

(6) 输入 `</video>` 结束 video 元素。

```
<body>
    <video controls="controls">
        <source src="paddle-steamer.mp4"
        ➝ type="video/mp4">
        <source src="paddle-steamer.webm"
        ➝ type="video/webm">
        <p>Sorry, your browser doesn't
        ➝ support the video element</p>
    </video>
</body>
```

图 17.8.1　这里为视频定义了两个源：一个 MP4 文件和一个 WebM 文件。旧的浏览器则只会显示 p 元素中的消息

17.9　多个媒体源和 source 元素

接下来我们将学习 source 元素的各种可用的属性，不过在此之前，不妨先简单地了解一下为同一媒体指定多个来源的工作原理。

当浏览器发现 video 元素时，首先会查看该元素本身是否定义了 src。如果没有，就会检查 srouce 元素。浏览器会逐个查看这些来源，直到找到它可以播放的一个来源。一旦找到这样一个，就会播放它并忽略其他的来源。

在前一个例子中，Safari 会播放 MP4 文

件并完全忽略 WebM 文件，而 Firefox 则无法播放 MP4 文件，从而转向它能播放的 WebM 文件，如图 17.9.1 所示。

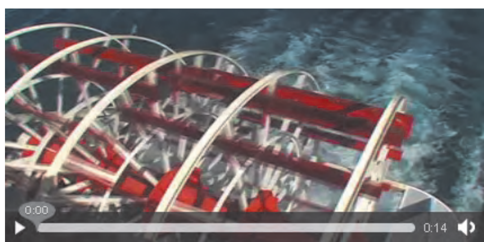


图 17.9.1　所有支持 HTML5 的浏览器都能加载该视频，因为我们为它指定 WebM 和 MP4 两种来源

既不识别 video 元素又不识别 source 元素的浏览器（即不支持 HTML5 的浏览器）在解析文档时会完全忽略这些标签，它只会显示在 video 元素结束标记之前输入的文本。

下面浏览一下 source 元素的属性（如表 17.2 所示）。

表 17.2　source 的属性

名　称	描　述
src	视频来源的 URL
type	用于指定视频的类型，帮助浏览器决定它是否能播放该视频。如 17.8 节中的例子所示，该属性的值反映的是视频的格式，即编解码器（如 video/mp4、video/webm、video/ogg）
media	用于为视频来源指定 CSS3 媒体查询，从而可以为具有不同屏幕尺寸的设备指定不同的（如更小的）视频

提示　如果在 video 元素本身的 src 属性中指定了值，它就会自动覆盖任何 source 元素中指定的来源。

17.10　添加具有备用超链接的视频

并非所有的浏览器都能播放 HTML5 视频（如 Internet Explorer 8 及以下的版本）。对于这些浏览器，需要提供后备方案。

或许对你来说，前一节介绍的示例中同时使用 video 和 source 元素的方式已经非常理想了。

你是对的。

无法理解 video 和 source 元素的浏览器会直接忽略它们，可以利用这一点实现后备方案。

在那个示例中，对于使用不支持 HTML5 的浏览器的用户，会显示一条文本消息。可以将这条消息替换为一个指向视频文件的超链接，从而让用户可以下载该文件并在闲暇时观看该视频。

在这个例子中（如图 17.10.1 和图 17.10.2 所示），我选择包含一个指向 MP4 版本视频的下载链接，不过提供指向 WebM 文件的链接或提供指向这两种格式文件的链接同样也很容易。

```
<body>
    <video controls="controls">
       <source src="paddle-steamer.mp4"
       ➝ type="video/mp4">
       <source src="paddle-steamer.webm"
       ➝ type="video/webm">
       <a href="paddle-steamer.mp4">
       ➝ Download the video</a>
    </video>
</body>
```

图 17.10.1　为视频指定了 MP4 和 WebM 两种来源，同时，在旧的浏览器中将显示下载该 MP4 文件的链接。



图 17.10.2　Internet Explorer 8 会忽略 video 和 source 元素，仅显示下载链接

为视频添加备用超链接的步骤

(1) 获取视频资源。

(2) 输入 <video controls="controls"> 开始 video 元素（含默认控件集）。

(3) 输入 <source src="*myVideo.mp4*"type="video/mp4">，这里的 *myVideo.mp4* 是 MP4 视频源文件的名称。

(4) 输入 <source src="*myVideo.webm*"type="video/webm">，这里的 *myVideo.webm* 是 WebM 视频源文件的名称。

(5) 输入 Download the video（这里的 *myVideo.mp4* 是视频源文件的名称）以指定备用视频文件超链接，用户可以通过这个超链接下载视频。

(6) 输入 </video> 结束 video 元素。

17.11　添加具有备用 Flash 的视频

除了可以提供下载链接作为后备方案，还可以（或许可以说应该）嵌入一个能播放 MP4 视频文件的 Flash 播放器。

是的，尽管 HTML5 和原生多媒体非常强大，但为了照顾那些无法处理这些技术的旧浏览器，恐怕还得求助于嵌入 Flash 内容的方法。也就是说，如果希望获得尽可能多的用户，至少还有一种选择！

过去，要在网页中嵌入备用 Flash 播放器和视频，既可以使用 object 元素，也可以使用 embed 元素，不过这两种方法都不是严格有效的 HTML，因为它们都没有出现在规范中。

HTML5 规范包含了 embed 和 object 元素，因此它们现在至少是有效的 HTML。

我们将使用 object 元素，因为该元素提供的解决方案更完整——包含在 object 元素中的内容都会呈现出来，哪怕浏览器并不支持 object 元素指定的插件。这样就可以指定另外一个后备方案了（也应该这样做），即后备方案中的后备方案。

此外，我推荐下载并使用开源的 Flash 视频播放器（如 JW Player、Flowplayer 等），它们可以让上述嵌入视频的过程变得相当容易，如图 17.11.1 所示。如果播放器能播放 MP4 文件，就可以对现有的视频源文件进行复用，这是理想情况；如果不能播放 MP4 文件，则需要将视频转化为 SWF 或 FLV 文件。

```
<body>
    <video controls="controls">
        <source src="paddle-steamer.mp4"
        ➝ type="video/mp4">
        <source src="paddle-steamer.webm"
        ➝ type="video/webm">
        <object type="application/
        ➝ x-shockwave-flash" data=
        ➝ "player.swf?videoUrl=paddle-
        ➝ steamer.mp4&controls=true">
            <param name="movie" value=
            ➝ "player.swf?videoUrl=paddle-
            ➝ steamer.mp4&controls=true" />
        </object>
    </video>
</body>
```

图 17.11.1　不支持 HTML5 的浏览器将转而使用备用 Flash 播放器，播放指定的 MP4 视频文件

备用 Flash 播放器

上面的代码片断中提到的备用 Flash 播放器（player.swf）包含在本章可供下载的代码中。这里用的播放器是相当出色的 JW Player（由 LongTail Video 出品；www.longtailvideo.com/players/jw-flv-player）。

1. 为视频添加备用 Flash 的步骤

(1) 获取视频资源。

(2) 输入 <video controls="controls"> 开始 video 元素（含默认控件集）。

(3) 输入 <source src="*myVideo.mp4*" type=

"video/mp4">，这里的 *myVideo.mp4* 是 MP4 视频源文件的名称。

(4) 输入 <source src="*myVideo.webm*"type="video/webm">，这里的 *myVideo.webm* 是 WebM 视频源文件的名称。

(5) 输入 <object type="application/x-shockwave-flash"data="*player.swf*?videoUrl=*myVideo.mp4*&controls=true">（这里的 *myVideo.mp4* 是视频源文件的名称）以指定这是一个备用 Flash 播放器，同时指定要使用的播放器和视频文件。注意这里指定的参数仅对本章所用的 *player.swf* 有效。

(6) 输入 <param name="movie" value="*player.swf*?videoUrl=*myVideo.mp4*&con-trols=true" />（这里的 *myVideo.mp4* 是视频源文件的名称），从而为不理解 object 元素开始标记中信息的浏览器指定要使用的播放器和视频文件。注意这里指定的参数仅对本章所用的 *player.swf* 有效。

(7) 输入 </object> 结束 object 元素。

(8) 输入 </video> 结束 video 元素。

```
<body>
    <video controls="controls">
        <source src="paddle-steamer.mp4"
        → type="video/mp4">
        <source src="paddle-steamer.webm"
        → type="video/webm">
        <object type="application/
        → x-shockwave-flash" data=
        → "player.swf?videoUrl=paddle-
        → steamer.mp4&controls=true">
            <param name="movie" value=
            → "player.swf?videoUrl=paddle-
            → steamer.mp4&controls=true" />
        </object>
        <a href="paddle-steamer.mp4">
        → Download the video</a>
    </video>
</body>
```

图 17.11.2 Internet Explorer 8 中的备用 Flash 播放器

此外，还可以在 Flash 对象后面、video 元素结束标记前面添加一个视频文件下载链接（如前面的示例那样）。这样就提供了进一步的后备方案，让用户可以下载视频文件，如图 17.11.2 和图 17.11.3 所示。不过，在不支持 HTML5 视频的浏览器中，备用 Flash 播放器会与下载链接一同显示出来。

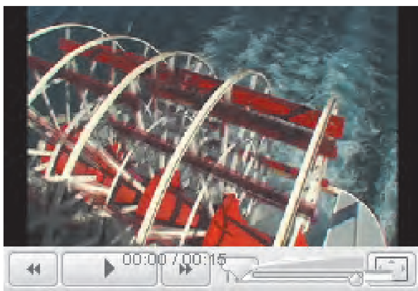


图 17.11.3 不支持 HTML5 视频的浏览器会转而使用备用 Flash 播放器，该播放器将播放指定的 MP4 视频文件。下载链接也会显示出来，没有安装 Flash 的浏览器会提供这种选择

2. 为视频添加备用 Flash 和超链接

(1) 获取视频资源。

(2) 输入 <video controls="controls"> 开始 video 元素（含默认控件集）。

(3) 输入 <source src="*myVideo.mp4*" type="video/mp4">，这里的 *myVideo.mp4* 是 MP4 视频源文件的名称。

(4) 输入<source src="*myVideo.webm*" type="video/webm">，这里的 *myVideo.webm* 是 WebM 视频源文件的名称。

(5) 输入<object type="application/x-shockwave-flash" data="*player.swf*?videoUrl=*myVideo.mp4*&controls=true">（这里的 *myVideo.mp4* 是视频源文件的名称）以指定这是一个备用 Flash 播放器，同时指定要使用的播放器和视频文件。注意这里指定的参数仅对本章所用的 *player.swf* 有效。

(6) 输入 `<param name="movie" value="player.swf?videoUrl=myVideo.mp4&controls=true" />`（这里的 *myVideo.mp4* 是视频源文件的名称），从而为不理解 object 元素开始标记中信息的浏览器指定要使用的播放器和视频文件。

(7) 输入 </object> 结束 object 元素。

(8) 输入 `<a href="myVideo.mp4">Download the video</a>`（这里的 *myVideo.mp4* 是视频源文件的名称）以指定备用视频文件链接。

(9) 输入 </video> 结束 video 元素。

提 示　如果浏览器支持 HTML5 视频，但无法找到它能播放的文件，它将不会使用备用 Flash 播放器，如图 17.11.4 所示。



图 17.11.4　Firefox 在找不到可播放的视频文件时显示的样子（具有默认控件集）——它不会使用备用 Flash 播放器，也不会显示下载链接

提 示　关于如何制作每个人都能访问的视频，Kroc Camen 提供了一份很好的材料，名为 "Video for Everybody"（http://camendesign.com/code/video_for_everybody）。此外，Jonathan Neal 提供的 Video for Everybody Generator 也很值得下载（http://sandbox.thewikies.com/vfe-generator/）。

17.12　提供可访问性

利用现代浏览器提供的原生可访问性支持，原生多媒体可以更好地使用键盘进行控制，这是原生多媒体的另一个好处。

你应该也会这么想！

Opera 是目前唯一的 HTML5 媒体默认控件集拥有键盘可访问性的现代浏览器。对于其他的浏览器，让媒体播放器变得键盘可访问的唯一途径是创建你自己的控件集。为此，需要使用 JavaScript Media API（这也是 HTML5 的一部分），不过这已经超出了本章的讨论范围。

HTML5 还指定了一种新的文件格式用于包含文本字幕、标题、描述、篇章等视频内容。

WebVTT（Web Video Text Track，Web 视频文本轨道）文件格式就是用于标记外部文本轨道资源（如字幕）的。目前还没有浏览器支持这种格式，但已经出现了大量可以处理 WebVTT 及其功能的 JavaScript 库（如 Playr 和 Captionator）。

关于 WebVTT 和字幕的进一步讨论已经超出了本章的范围，不过更多信息可以参见 www.iandevlin.com/blog/2011/05/html5/webvtt-and-video-subtitles。

提 示　Ian Devlin 的 *HTML5 Multimedia: Develop and Design*（Peachpit Press，2011）一书中有专门演示如何创建个人的可访问控件集和使用 WebVTT 的章节。

17.13　添加音频文件格式

学会了使用 HTML5 原生媒体在网页中添加视频之后，下面看看如何添加音频。同 HTML5 视频一样，HTML5 也支持大量不同的音频文件格式（编解码器）。

可供使用的音频编解码器主要有五种。以下是这五种编解码器及支持它们的浏览器。

- Ogg Vorbis 使用的文件扩展名为 .ogg，支持它的浏览器包括 Firefox 3.5+、

Chrome 5+ 和 Opera 10.5+。

- MP3 使用的文件扩展名为 .mp3，支持它的浏览器包括 Safari 5+、Chrome 6+、Internet Explorer 9+ 和 iOS。
- WAV 使用的文件扩展名为 .wav，支持它的浏览器包括 Firefox 3.6+、Safari 5+、Chrome 8+ 和 Opera 10.5+。
- AAC 使用的文件扩展名为 .aac，支持它的浏览器包括 Safari 3+、Internet Explorer 9+、iOS 3+ 和 Android 2+。
- MP4 使用的文件扩展名为 .mp4，支持它的浏览器包括 Safari 3+、Chrome 5+、Internet Explorer 9+、iOS 3+ 和 Android 2+。

你应该还记得，MP4 是一种视频编解码器，不过它也可以仅对音频数据进行编码。

提示　对于视频，需要使用两种不同的文件格式才能确保获得所有兼容 HTML5 的浏览器的支持。对于音频，最好的两种格式是 Ogg Vorbis 和 MP3。

提示　在“转换文件格式”中提到的 Miro Video Converter 应用程序也可以用来转化音频。

17.14　在网页中添加单个音频文件

下面开始讲解如何在网页中放置音频文件。这一过程同添加视频的过程是相似的，不过这次用到的是 audio 元素，如图 17.14.1 所示。

```
<body>
    <audio src="piano.ogg"></audio>
</body>
```

图 17.14.1　单个 Ogg 编码的音频（不含控件）

在网页中添加单个音频文件的步骤

(1) 获取音频文件。

(2) 输入 `<audio src="myAudio.ext"></audio>`，这里的 *myAudio.ext* 是音频文件的位置、名称和扩展名。

17.15　在网页中添加带控件的单个音频文件

如前面的例子所示，在网页中添加单个音频文件是很容易的。不过仅仅这样做的话，什么也不会显示，因为音频文件不可见，因此需要使用 controls 属性添加一些控件，如图 17.15.1 所示。

```
<body>
    <audio src="piano.ogg" controls=
    ➝ "controls"></audio>
</body>
```

图 17.15.1　单个 Ogg 编码的音频（包含默认控件集）

在网页中添加带控件的单个音频文件的步骤

(1) 获取音频文件。

(2) 输入 `<audio src="myAudio.ext" controls="controls"></audio>`。

当然，同视频控件一样，每个浏览器都有处理控件外观的独特方式（如图 17.15.2 ~ 图 17.15.6 所示）。



图 17.15.2　Firefox 中的音频控件

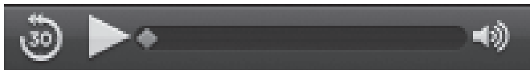


图 17.15.3　Safari 中的音频控件



图 17.15.4　Chrome 中的音频控件



图 17.15.5　Opera 中的音频控件



图 17.15.6　Internet Explorer 9 中的音频控件

17.16　音频属性一览

同 video 元素一样，audio 元素也有很多可用的属性，如表 17.3 所示。

表 17.3　音频属性

名　称	描　述
src（源）	指定音频文件的 URL
autoplay（自动播放）	当音频可以播放时立即开始播放
controls（控件）	添加浏览器为音频设置的默认控件
muted（静音）	让音频静音（目前没有任何浏览器支持）
loop（循环）	让音频循环播放
preload（预加载）	告诉浏览器要加载的音频内容的多少。可以是以下三个值： none 表示不加载任何音频 metadata 表示仅加载音频的元数据（如长度） auto 表示让浏览器决定怎样做（这是默认的设置）

17.17　为音频添加控件、自动播放并设置循环播放

使用 controls 和 autoplay 属性为音频文件添加控件并指定其在加载后自动播放相当容易，如图 17.17.1 和图 17.17.2 所示。

```
<body>
    <audio src="piano.ogg" autoplay=
    ➝ "autoplay" controls="controls">
    ➝ </audio>
</body>
```

图 17.17.1　当页面加载时会自动播放的 Ogg 音频文件（含默认控件集）



图 17.17.2　加载时会自动开始播放的音频文件（含控件）

还可以使用 loop 属性让音频文件循环播放，如图 17.17.3 所示。

```
<body>
    <audio src="piano.ogg" loop="loop"
    ➝ controls="controls"></audio>
</body>
```

图 17.17.3　会循环播放的 Ogg 音频文件（含默认控件集）

1. 为音频文件添加控件并让其自动播放的步骤

(1) 获取音频文件。

(2) 输入 <audio src="*myAudio.ext*" autoplay="autoplay" controls="controls"></audio>，这里的 *myAudio.ext* 是音频文件的位置、名称和扩展名。

2. 让音频文件循环播放的步骤

(1) 获取音频文件。

(2) 输入 <audio src="*myAudio.ext*" loop="loop" controls="controls"></audio>，这里的 *myAudio.ext* 是音频文件的位置、名称和扩展名。

提示　Firefox 不支持 loop 属性。

提示　可以让音频自动播放和循环播放并不意味着这样做是必需的。

17

17.18　预加载音频文件

通过使用表 17.3 所列的不同的 audio 元素属性，可以以不同的方式请求浏览器对音频文件进行预加载，如图 17.18.1 和图 17.18.2 所示。

```
<body>
    <audio src="piano.ogg" preload=
    → "metadata" controls="controls">
    → </audio>
</body>
```

图 17.18.1　当页面加载时，这个 Ogg 音频文件仅包含元数据（如长度）

```
<body>
    <audio src="piano.ogg" preload="auto"
    → controls="controls"></audio>
</body>
```

图 17.18.2　这个 Ogg 音频文件让浏览器决定要加载的内容的多少

1. 让浏览器仅预加载音频元数据

(1) 获取音频文件。

(2) 输入 <audio src="*myAudio.ext*" preload="metadata" controls="controls"></audio>，这里的 *myAudio.ext* 是音频文件的位置、名称和扩展名。

2. 让浏览器决定如何预加载音频文件

(1) 获取音频文件。

(2) 输入 <audio src="*myAudio.ext*" preload="auto" controls="controls"></audio>，这里的 *myAudio.ext* 是音频文件的位置、名称和扩展名。

提示　指定 preload 属性的值并不能确保浏览器会照做，只能作为一种请求。

提示　通过 autoplay 属性指定音频文件自动播放会覆盖 preload 属性的设置，因为音频文件必须加载以后才能播放。

17.19　提供多个音频源

如前所述，为了获得所有兼容 HTML5 的浏览器的支持，至少需要为音频提供两种格式。实现这一目标的方法同 video 元素也是一样的，即使用 source 元素，如图 17.19.1 所示。

```
<body>
    <audio controls="controls">
        <source src="piano.ogg" type=
        → "audio/ogg">
        <source src="piano.mp3" type=
        → "audio/mp3">
    </audio>
</body>
```

图 17.19.1　这个 audio 元素（含默认控件集）定义了两个音频源文件，一个编码为 Ogg，另一个为 MP3

完整的过程同指定多个视频源文件的过程是一样的。浏览器会忽略它不能播放的，仅播放它能播放的。

指定两种不同的音频来源的步骤

(1) 获取音频文件。

(2) 输入 <audio controls="controls"> 开始 audio 元素（含默认控件集）。

(3) 输入 <source src="*myAudio.ogg*" type="audio/ogg">，这里的 *myAudio.ogg* 是 Ogg Vorbis 音频文件的位置、名称和扩展名。

(4) 输入 <source src="*myAudio.mp3*" type="audio/mp3">，这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名。

(5) 输入 </audio> 结束 audio 元素。

提示　type 属性可以帮助浏览器判断它是否能播放某个文件。对音频文件来说，其值总是 audio/ 加上格式本身，包括 audio/ogg、audio/mp3、audio/aac、audio/wav 和 audio/mp4。

17.20 添加具有备用超链接的音频

为音频建立后备方案的方法同视频的情况是完全相同的，因此我在这里免不了要重复了。

可以使用 audio 和 source 元素定义多个来源，并在 audio 元素结束之前为不支持 HTML5 的浏览器添加后备方案，如图 17.20.1 所示。

```
<body>
    <audio controls="controls">
        <source src="piano.ogg" type=
        ➝ "audio/ogg">
        <source src="piano.mp3" type=
        ➝ "audio/mp3">
    <a href="piano.mp3">Download the
    ➝ audio</a>
    </audio>
</body>
```

图 17.20.1 这个 audio 元素定义了两个音频源文件，不支持 HTML5 的浏览器仅会显示用于下载该音频的 MP3 版本的超链接

为音频添加备用超链接的步骤

(1) 获取音频文件。

(2) 输入 <audio controls="controls"> 开始 audio 元素（含默认控件集）。

(3) 输入 <source src="*myAudio.ogg*" type="audio/ogg">，这里的 *myAudio.ogg* 是 Ogg Vorbis 音频文件的位置、名称和扩展名。

(4) 输入 <source src="*myAudio.mp3*" type="audio/mp3">，这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名。

(5) 输入 Download the audio（这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名），为不支持 HTML5 的浏览器提供下载音频文件的超链接。

(6) 输入 </audio> 结束 audio 元素。

17.21 添加具有备用 Flash 的音频

同视频一样，Flash 也常被用做嵌入音频内容的插件。同样，也可以为不支持 HTML5 的浏览器（如 Internet Explorer 8）提供备用 Flash 播放器。

为音频添加备用 Flash 的步骤

(1) 获取音频文件。

(2) 输入 <audio controls="controls"> 开始 audio 元素（含默认控件集）。

(3) 输入 <source src="*myAudio.ogg*" type="audio/ogg">，这里的 *myAudio.ogg* 是 Ogg Vorbis 音频文件的位置、名称和扩展名。

(4) 输入 <source src="*myAudio.mp3*" type="audio/mp3">，这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名。

(5) 输入 <object type="application/xshockwave-flash" data="*player.swf*?audioUrl=*myAudio.mp3*&controls=true">（这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名）以指定这是一个备用 Flash 播放器，同时指定要使用的播放器和音频文件。在这个例子中，*player.swf* 同视频部分所用的备用 Flash 播放器是同一个文件。注意这里指定的参数仅对本章所用的 *player.swf* 有效。

(6) 输入 <param name="movie" value="*player.swf*?audioUrl=*myAudio.mp3*&controls=true" />（这里的 *myAudio.mp3* 是音频文件的位置、名称和扩展名），从而为不理解 object 元素开始标记中信息的浏览器指定要使用的播放器和音频文件。

(7) 输入 </object> 结束 object 元素。

(8) 输入 </audio> 结束 audio 元素，如图 17.21.1 和图 17.21.2 所示。

17

```
<body>
    <audio controls="controls">
        <source src="piano.ogg" type=
        ➝ "audio/ogg">
        <source src="piano.mp3" type=
        ➝ "audio/mp3">
        <object type="application/
        ➝ x-shockwave-flash"
            data="player.swf?audioUrl=
            ➝ piano.mp3&controls=true">
            <param name="movie" value=
            ➝ "player.swf?audioUrl=
            ➝ piano.mp3&controls=true" />
        </object>
    </audio>
</body>
```

图 17.21.1 这个 audio 元素定义了两个音频来源，而 Internet Explorer 8 等浏览器将转而使用指定的备用 Flash 播放器（该播放器使用 MP3 文件作为音频来源）



图 17.21.2 Internet Explorer 8 中的音频备用 Flash 播放器

提 示 Internet Explorer 8 等浏览器会忽略 audio 和 source 元素，直接使用备用 Flash 播放器。只要用户安装了 Flash，就能播放音频内容。

17.22 添加同时具有备用 Flash 和超链接的音频

可以在备用 Flash 播放器后面添加一个下载链接，从而提供额外的后备方案，如图 17.22.1 和图 17.22.2 所示。

为音频添加备用 Flash 和超链接的步骤

(1) 获取音频文件。

(2) 输入 <audio controls="controls"> 开始 audio 元素（含默认控件集）。

(3) 输入 <source src="*myAudio.ogg*" type="audio/ogg">，这里的 *myAudio.ogg* 是 Ogg Vorbis 音频文件的位置、名称和扩展名。

(4) 输入 <source src="*myAudio.mp3*" type="audio/mp3">，这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名。

(5) 输入 Download the audio（这里的 *myAudio.mp3* 是 MP3 音频文件的位置、名称和扩展名），为不支持 HTML5 的浏览器提供下载音频文件的超链接。

```
<body>
    <audio controls="controls">
        <source src="piano.ogg" type=
        ➝ "audio/ogg">
        <source src="piano.mp3" type=
        ➝ "audio/mp3">
        <object type="application/
        ➝ x-shockwave-flash" data=
        ➝ "player.swf?audioUrl=piano.mp3
        ➝ &controls=true" width="280">
            <param name="movie" value=
            ➝ "player.swf?audioUrl=piano.mp3
            ➝ &controls=true" />
        </object>
        <a href="piano.mp3">Download the
        ➝ audio</a>
    </audio>
</body>
```

图 17.22.1 这里为 HTML5 浏览器定义了两个音频源文件，为支持 Flash 的浏览器（如 Internet Explorer 8）定义了一个备用 Flash 播放器。此外，还通过一个简单的 MP3 音频文件链接添加了一个额外的后备方案



图 17.22.2 Internet Explorer 8 中的音频备用 Flash 和超链接

(6) 输入 <object type="application/x-shockwave-flash" data="*player.swf*?audioUrl=*myAudio.mp3*&controls=true">（这里的 *myAudio*

.*mp3* 是 MP3 音频文件的位置、名称和扩展名）以指定这是一个备用 Flash 播放器，同时指定要使用的播放器和音频文件。注意这里指定的参数仅对本章所用的 *player.swf* 有效。

(7) 输入 `<param name="movie" value="player.swf?audioUrl=myAudio.mp3&controls=true" />`（这里的 *myAudio.mp3* 是音频文件的位置、名称和扩展名），从而为不理解 `object` 元素开始标记中信息的浏览器指定要使用的播放器和音频文件。

(8) 输入 `</object>` 结束 `object` 元素。

(9) 输入 `<a href="myAudio.mp3">Download the audio</a>`。

(10) 输入 `</audio>` 结束 `audio` 元素。

17.23　获取多媒体文件

音频和视频是嵌入到网页中的最常见的多媒体文件。可以使用麦克风和数字化软件（如 Windows 上的 Sound Recorder、Mac 上的 Amadeus 等）创建音频。此外，还有很多程序可以用于从 CD 创建 MP3。

随着智能手机及其摄像头（正在不断提升）的发展，在万维网上获得视频变得更加容易了。即便视频的格式并不是你需要的，仍然可以通过 Miro Video Converter、HandBrake 等软件对格式进行转换，这个过程相当容易。

还可以在万维网上查找声音和电影文件，但是应该仔细地阅读相应的许可协议。

不过，不要局限于使用音频和视频。尽管使用 HTML5 的 `canvas` 元素及其 JavaScript API 可以创建动画，但仍可以像以前那样嵌入 Flash 动画（即使用 `object` 元素）。HTML5 媒体在不断增强，但 Flash 仍然有它的位置。

17.24　考虑数字版权管理（DRM）

在嵌入音频和视频文件的过程中，你一定注意到了这样一个事实——所有人都看得见指向源文件的 URL，从而利用它下载并“盗取”你的内容，就像嵌入的图像以及 HTML、JavaScript 和 CSS 源文件一样。

对此我们毫无办法，因为 HTML5 并没有提供任何保护媒体内容的方法（不过在将来有可能出现）。

因此，如果你很关心对媒体文件的保护，那么暂时不要使用本章讲解的 HTML5 原生多媒体和备用 Flash 方法，因为 DRM 需要嵌入媒体文件，且 DRM 工具已经烧录进了源材料。

17.25　嵌入 Flash 动画

Adobe Flash 软件可以创建动画、电影以及其他广泛用于万维网的媒体。与之相伴的插件通常用于在网页中嵌入视频和音频。但能用到 Flash 的地方还不止这些。有很多动画是使用 Adobe Flash 制作的，尽管它们无法在 iPad 和 iPhone 这样的设备上播放，但仍然有用得到它们的时候。

前面已经介绍了如何使用 Adobe Flash 嵌入音频和视频，从而为含 Flash 播放器的旧浏览器提供后备方案。接下来将要介绍如何嵌入真正的 Adobe Flash 动画 SWF 文件（如图 17.25.1 和图 17.25.2 所示）。

嵌入 Flash 动画的步骤

(1) 输入 `<object` 开始 `object` 元素。

(2) 输入 `type="application/x-shockwave-flash"`，说明 MIME 类型为 Flash 动画。

(3) 输入 `data="filename.swf"`，这里的 *filename.swf* 是 Flash 动画在服务器上的名称和位置。

(4) 输入 width="*w*" height="*h*" 指定动画的尺寸，这里的 *w* 和 *h* 是以像素为单位的值。

(5) 输入 > 结束 object 的开始标记。

(6) 输入 <param name="movie" value="*filename.swf*" />，这里的 *filename.swf* 与第 (3) 步中使用的是同一个文件。

(7) 输入 </object> 完成这个对象。

```
<head>
<title>Embed Flash Movie</title>
</head>
<body>
<object type="application/x-shockwave-
→flash" data="http://www.sarahsnotecards
→.com/catalunyalive/minipalau.swf"
→width="300" height="240">
<param name="movie" value="http://
→www.sarahsnotecards.com/catalunyalive/
→minipaulau.swf" />
</object>
</body>
```

图 17.25.1　要嵌入 Flash 动画，将 MIME 类型设为 application/x-shockwave-flash

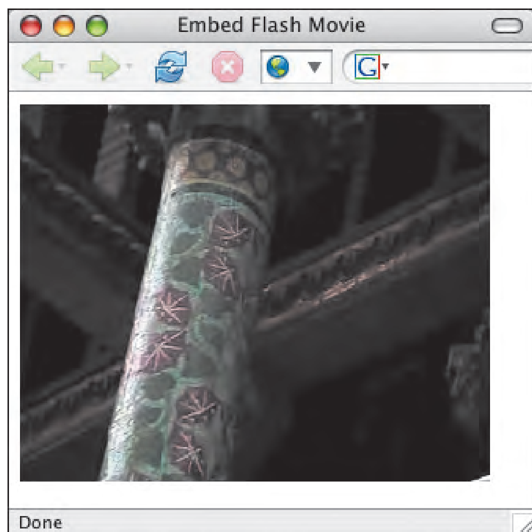


图 17.25.2　使用 object 元素在网页中嵌入了 Flash 动画

提示　这种技术基于 Drew McLellan 在 *A List Apart* 上发表的文章“Flash Satay”（www.alistapart.com/articles/flashsatay）。

提示　Drew 找到了通过这种技术使用小的引用电影帮助 Flash 动画流正确工作的一种途径。详情参见他的文章。

提示　很多人组合使用 object 和 embed 标签在网页中插入 Flash 动画，这两种标签都是有效的 HTML5 标记。要查看更多信息，可以在 Adobe 公司的网站（www.adobe.com）上搜索 embed Flash。

17.26　嵌入 YouTube 视频

YouTube（以及其他类似的服务）提供了可以上传视频文件的服务器（这些文件的数量已经非常庞大了），并让访问者可以观看这些视频。

嵌入 YouTube 视频的步骤

(1) 访问 YouTube 网站（www.youtube.com）并找到你想使用的视频。

(2) 从地址栏复制视频代码，它在 v= 之后，一直到第一个 & 符号为止。

(3) 按照前一节中“嵌入 Flash 动画的步骤”，嵌入 Flash。在必须插入 Flash 动画的 URL 的两个位置，输入 http://www.youtube.com/v/*moviecode*，这里的 *moviecode* 是第 (2) 步中从地址栏复制的代码。

提示　获取 YouTube 视频代码时，代码出现在 v= 的后面；编写引用视频的 URL 时，使用的是 v/。

17.27　通过 canvas 操作视频

使用 HTML5 原生多媒体的另一个好处是

可以利用很多来自 HTML5 或与 HTML5 相关的新特性和新功能。

这些新特性中的一个便是 canvas（画布）元素。

使用 canvas 元素及相应的 JavaScript API 可以在网页上描制并创建动画。

可以对 HTML5 视频应用这些 API，因为 video 元素可以同其他 HTML 元素一样进行处理，因此它也可以被 canvas 访问和获取。

通过 JavaScript API，可以从播放的视频中抓取图像，并在 canvas 元素中重新绘制该图像，从而创建视频的截图。

通过 API 可以对单个图像像素进行操作，同时由于可以根据视频在 canvas 中创建图像，因而这意味着可以调整视频的像素。例如，可以将视频转化为灰度模式。

这只是让你对通过 canvas 操作 video 元素建立一些简单的概念，对这一主题的深入探讨已经超出了本书的范围。更多关于 canvas 及其 JavaScript API 的信息，参见 17.29 节。

17.28　联合使用 SVG 和视频

人们开始关注的另一项与 HTML5 有关的技术是 SVG（Scalable Vector Graphic，可缩放矢量图形）。

SVG 已经存在相当一段时间了（它诞生自 1999 年），但直到 HTML5 才有了 svg 元素。通过该元素可以在网页本身嵌入 SVG 定义。

SVG 使用 XML 定义图形和图像，浏览器则对其进行解释和使用，从而描绘出真正的图形。SVG 定义所包含的全部内容就是对如何绘制和绘制什么的说明。

使用 SVG 创建的图像也是基于矢量而不是基于光栅的。这意味着它们可以很好地适应缩放，因为浏览器只是简单地依照绘制说明，根据所需的尺寸，将图形绘制出来。光栅图像包含的是像素数据，如果要以远大于原始图像的尺寸重新绘制图像，就会因为缺少足够的像素数据导致图像质量受损。

关于 SVG 的完整讨论超出了本章的范围，这里只是想让你知道视频可以同 SVG 定义联合使用。通过 SVG 创建的图形可以用于对视频进行遮罩，也即只显示能透过该图形（如圆圈）的底层视频。

此外，还有一些 SVG 滤镜可以应用于 HTML5 视频，如黑白转换、高斯模糊、色彩饱和度等。关于 SVG 的更多信息，参见 17.29 节。

17.29　更多资源

本章仅涵盖了 HTML5 多媒体的基础知识。关于这一主题，还有很多值得学习的东西。这里有大量有关资源供你研究。

1. 在线资源

- “Video on the Web”（http://diveinto.html5doctor.com/video.html）
- *HTML5 Video*（http://html5video.org）
- “WebVTT and Video Subtitles”（www.iandevlin.com/blog/2011/05/html5/webvtt-and-video-subtitles）
- “HTML5 Canvas: The Basics”（http://dev.opera.com/articles/view/html-5-canvas-the-basics）
- “Learning SVG”（http://my.opera.com/tagawa/blog/learning-svg）

2. 图书

- Ian Devlin 所著的 *HTML5 Multimedia: Develop and Design*（Peachpit Press，2011），参见 http://html5multimedia.com
- Shelley Powers 所著的 *HTML5 Media*（O’Reilly Media，2011）
- Silvia Pfeiffer 所著的 *The Definitive Guide to HTML5 Video*（Apress，2010）

第 18 章

表　　格

18

本章内容

- ❑ 结构化表格
- ❑ 让单元格跨越多列或多行

在日常生活中，我们对表格式数据已经很熟悉了。这种数据有多种形式，如财务数据、调查数据、事件日历、公交车时刻表、电视节目表等。在大多数情况下，这类信息都由列标题或行标题加上数据本身构成。

本章将对 table 元素及其子元素进行讲解，重点是基本的 table 结构和样式。HTML 表格可以很复杂，不过很少需要实现特别复杂的表格（除非是具有丰富数据的网站）。下列链接展示了一些复杂表格结构的代码示例，并着重强调了如何提升表格的可访问性。

- ❑ Roger Johansson 的“Bring On the Tables”（www.456bereastreet.com/archive/200410/bring_on_the_tables/）。
- ❑ Roger Hudson 的“Accessible Data Tables”（www.usability.com.au/resources/tables.cfm）。
- ❑ Stephen Ferg 的“Techniques for Accessible HTML Tables”（www.ferg.org/section508/accessible_tables.html）。

18.1　结构化表格

放在电子表格中的信息通常很适合用 HTML 表格呈现。

从基本层面看，table 元素是由行组成的，行又是由单元格组成的。每个行（tr）都包含标题单元格（th）或数据单元格（td），或者同时包含这两种单元格。如果认为为整个表格添加一个标题有助于访问者理解该表格，可以提供 caption。此外，scope 属性（也是可选的，不过推荐使用）可以告诉屏幕阅读器和其他辅助设备当前的 th 是列的标题单元格（使用 scope="col"）还是行的标题单元格（使用 scope="row"），抑或是用于其他目的的单元格（参见最后一条提示）表格元素的基本实现代码如图 18.1.1 所示。

在默认情况下，表格在浏览器中呈现的宽度是其中的信息在页面可用空间里所需要的最小宽度，如图 18.1.2 所示。很容易就会想到可以通过 CSS 改变表格的格式，这很快就会讲到。

不过，图 18.1.1 中的表格似乎缺少一些东西。如何知道每行数据表示的是什么呢？如果每个行也有标题单元格，就很容易理解了。添加这些单元格只需要在每行开头添加一个 th 元素就可以了。列标题应设置 scope="col"，而每个行的 th（位于 td 之前）

则应设置 scope="row"，如图 18.1.3 所示。

```
...
<body>
<table>
    <caption>Quarterly Financials for
    → 1962-1964 (in Thousands)</caption>
    <tr>
       <th scope="col">1962</th>
       <th scope="col">1963</th>
       <th scope="col">1964</th>
    </tr>
    <tr>
       <td>$145</td>
       <td>$167</td>
       <td>$161</td>
    </tr>
    <tr>
       <td>$140</td>
       <td>$159</td>
       <td>$164</td>
    </tr>
    <tr>
       <td>$153</td>
       <td>$162</td>
       <td>$168</td>
    </tr>
    <tr>
       <td>$157</td>
       <td>$160</td>
       <td>$171</td>
    </tr>
</table>
</body>
</html>
```

图 18.1.1　每个行都是由 tr 元素标记的。这是个很简单的表格，它只有一个包含标题单元格（th 元素）的行和三个包含数据单元格（td 元素）的行。如果要包含 caption，它必须是 table 里的第一个元素（caption 还可以包含 p 和其他的文本元素）

图 18.1.2　在默认情况下，th 文本是以粗体显示的，th 和 caption 文本都是居中对齐的，表格的宽度就是内容所需的宽度

```
...
<body>
<table>
    <caption>Quarterly Financials for
    → 1962-1964 (in Thousands)</caption>
    <thead> <!-- 表格头部 -->
       <tr>
          <th scope="col">Quarter</th>
          <th scope="col">1962</th>
          <th scope="col">1963</th>
          <th scope="col">1964</th>
       </tr>
    </thead>
    <tbody> <!-- 表格主体 -->
       <tr>
          <th scope="row">Q1</th>
          <td>$145</td>
          <td>$167</td>
          <td>$161</td>
       </tr>
       <tr>
          <th scope="row">Q2</th>

          <td>$140</td>
          <td>$159</td>
          <td>$164</td>
       </tr>
       ... Q3 and Q4 rows ...
    </tbody>
    <tfoot> <!-- 表格尾部 -->
       <tr>
          <th scope="row">TOTAL</th>
          <td>$595</td>
          <td>$648</td>
          <td>$664</td>
       </tr>
    </tfoot>
</table>
</body>
</html>
```

图 18.1.3　通过指定 thead、tbody 和 tfoot 显式地定义了表格的不同部分。接着，在每行的开头添加了 th 元素。tbody 和 tfoot 中的 th 设置了 scope="row"，表明它们是行标题

图 18.1.3 中还特意引入了其他一些专门用于定义表格的元素，即 thead、tbody 和 tfoot。thead 元素可以显式地将一行或多行标题标记为表格的头部。tbody 元素用于包围所有的数据行。tfoot 元素可以显式地将一行

或多行标记为表格的尾部。可以使用 tfoot 包围对列的计算值（就像图 18.1.3 中所示的那样），也可以在长表格（如列车时刻表）中使用 tfoot 重复 thead 中的内容（如果表格在打印时超过一页，有的浏览器会在每一页都打印 tfoot 和 thead 元素的内容）。thead、tfoot 和 tbody 元素不会影响表格的布局，也不是必需的（不过推荐使用它们）。如果包含了 thead 或 tfoot，则必须同时包含 tbody。此外，还可以对它们添加样式。

如图 18.1.2 所示，在默认情况下，表格被压得很扁。通过应用一些 CSS（如图 18.1.4 所示），可以为单元格添加一些空间让它们扩大一些（使用 padding），添加边框以指示单元格的边界（使用 border），还可以对文本进行格式化。这些样式都有助于更好地理解表格，如图 18.1.5 所示。

创建表格结构的步骤

(1) 输入 <table>。

(2) 如果需要，输入 <caption>*caption content*</caption>，其中的 *caption content* 是对表格的描述。

(3) 如果需要，在要创建的表格部分的第一个 tr 元素之前，输入 <thead>、<tbody> 或 <tfoot>。

(4) 输入 <tr> 定义行的开始。

(5) 输入 <th scope="*scopetype*"> 开始标题单元格（其中的 *scopetype* 可以是 col、row、colgroup 或 rowgroup），或者输入 <td> 定义数据单元格的开始。

(6) 输入单元格的内容。

(7) 输入 </th> 结束标题单元格，或者输入 </td> 结束数据单元格。

(8) 对行内的每个单元格重复第 (5) 步至第 (7) 步。

(9) 输入 </tr> 结束行。

(10) 为所在表格部分内的每个行重复第 (4) 步至第 (9) 步。

(11) 如果在第 (3) 步中开始了一个表格部分，就根据需要使用 </thead>、</tbody> 或 </tfoot> 结束这个部分。

(12) 为每个表格部分重复第 (3) 步至第 (11) 步。注意，一个表格只能有一个 thead 和 tfoot，但可以有多个 tbody 元素。

(13) 输入 </table> 以结束表格。

```
body {
    font: 100% arial, helvetica, serif;
}

table {
    border-collapse: collapse;
}

caption {
    font-size: .8125em;
    font-weight: bold;
    margin-bottom: .5em;
}

th,
td {
    font-size: .875em;
    padding: .5em .75em;
}

td {
    border: 1px solid #000;
}

tfoot {
    font-style: italic;
    font-weight: bold;
}
```

图 18.1.4 这个简单的样式表为每个数据单元格添加了 border，为标题单元格和数据单元格都添加了 padding，还对表格 caption 和内容进行了格式化。如果不对 table 定义 border-collapse: collapse;，每个 td 的边框和相邻 td 的边框之间就会出现一些空隙（默认值为 border-collapse: separate;）。还可以像 18.2 节所示的那样，对 th 元素应用边框

Basic table evolved - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Basic table evolved

Quarterly Financials for 1962-1964
(in Thousands)

Quarter	1962	1963	1964
Q1	$145	$167	$161
Q2	$140	$159	$164
Q3	$153	$162	$168
Q4	$157	$160	$171
TOTAL	*$595*	*$648*	*$664*

图 18.1.5 现在，表格既有列标题，也有行标题，还有一个显示各列数据之和的行（位于 tfoot 元素中）。此外，这里还显示了为边框、单元格内边距、caption 内容和 tfoot 内容添加的样式

提 示 如果 table 是嵌套在 figure 元素内除 figcaption 以外的唯一元素，则可以省略 caption，使用 figcaption 对表格进行描述（参见 4.3 节）。注意，不要在 table 中嵌套 figcaption，而应跟往常一样将 figcaption 放在 figure 中。

提 示 还可以在样式表中对 table、td 或 th 元素设置 background、width 等属性，尽管没有在 CSS 示例中进行演示，如图 18.1.4 所示。总之，大多数用于其他 HTML 元素的文本格式和其他格式也可以应用于表格（参见 18.2 节的另一个例子）。你可能会发现，不同的浏览器显示的样式稍有差异，尤其是 Internet Explorer。

提 示 可以通过 scope 属性指定 th 为一组列的标题（使用 scope="colgroup"），或者为一组行的标题（使用 scope="rowgroup"）。对于后者，参见下一节的示例。

18.2 让单元格跨越多列或多行

可以通过 colspan 和 rowspan 属性让 th 或 td 跨越一个以上的列或行。对该属性指定的数值表示的是跨越的单元格的数量，如图 18.2.1 和图 18.2.2 所示。

```
...
<body>
<table>
    <caption>TV Schedule</caption>
    <thead> <!-- 表格头部 -->
       <tr>
          <th scope="rowgroup">Time</th>
          <th scope="col">Mon</th>
          <th scope="col">Tue</th>
          <th scope="col">Wed</th>
       </tr>
    </thead>
    <tbody> <!-- 表格主体 -->
       <tr>
          <th scope="row">8 pm</th>
          <td>Staring Contest</td>
          <td colspan="2">Celebrity Hoedown
          → </td>
       </tr>
       <tr>
          <th scope="row">9 pm</th>
          <td>Hardy, Har, Har</td>
          <td>What's for Lunch?</td>
          <td rowspan="2">Movie of the Week
          → </td>
       </tr>
       <tr>
          <th scope="row">10 pm</th>
          <td>Healers, Wheelers &amp;
          → Dealers</td>
          <td>It's a Crime</td>
       </tr>
    </tbody>
</table>
</body>
</html>
```

图 18.2.1 通过在包含 *Celebrity Hoedown* 演出的 td 上应用 colspan="2"，指示该演出的上映时间包含周二和周三晚上 8 点这两个时间。类似地，在包含 *Movie of the Week* 的 td 上添加了 rowspan="2"，因为该演出持续两个小时。同时，注意 Time 这个 th 设置了 scope="rowgroup"，因为它是它正下方的每个行标题组的标题

Table with colspan and rowspan - Mozilla Firefox

TV Schedule

TIME	MON	TUE	WED
8 pm	Staring Contest	Celebrity Hoedown	
9 pm	Hardy, Har, Har	What's for Lunch?	Movie of the Week
10 pm	Healers, Wheelers & Dealers	It's a Crime	

图 18.2.2　仅仅通过查看代码，很难判断 `colspan` 和 `rowspan` 对表格的影响，但在浏览器中查看就清楚多了。这个表格还用 CSS 添加了一些样式，完整的样式表见本书的配套网站（www.bruceontheloose.com/htmlcss/examples/）

1. 让单元格跨越两个或两个以上列的步骤

(1) 在需要定义跨越一个以上的列的单元格的地方，输入 `<td` 和一个空格。

(2) 输入 `colspan="n">`，这里的 *n* 是单元格要跨越的列数。

(3) 输入单元格的内容。

(4) 输入 `</td>`。

(5) 根据 18.1 节中的介绍，完成表格的其余部分。如果创建了一个跨越两列的单元格，在该行就应该少定义一个单元格；如果创建了一个跨越三列的单元格，在该行就应该少定义两个单元格，以此类推。

2. 让单元格跨越两个或两个以上行的步骤

(1) 在需要定义跨越一个以上的行的单元格的地方，输入 `<td` 和一个空格。

(2) 输入 `rowspan="n">`，这里的 *n* 是单元格要跨越的行数。

(3) 输入单元格的内容。

(4) 输入 `</td>`。

(5) 根据“结构化表格”一节中的介绍，完成表格的其余部分。如果创建了一个 `rowspan` 等于 2 的单元格，就不需要定义下一行中该单元格对应的单元格了；如果创建了一个 `rowspan` 等于 3 的单元格，就不需要定义下面两行中该单元格对应的单元格了，以此类推。

提 示　表格中的每一行都应具有相同的单元格数量。跨越多列的单元格应算做多个单元格，它的 `colspan` 属性值为多少，就算做多少个单元格。

提 示　表格中的每一列都应具有相同的单元格数量。跨越多行的单元格应算做多个单元格，它的 `rowspan` 属性值为多少，就算做多少个单元格。

第 19 章

处理脚本

19

本章内容

- 加载外部脚本
- 添加嵌入脚本
- 处理 JavaScript 事件

HTML 定义网页的内容，CSS 定义网页的表现，JavaScript 则定义特殊的行为。

通过编写简单的 JavaScript 程序，可以显示和隐藏内容；通过编写复杂一些的程序，可以加载数据并动态地更新页面。可以操作定制的 HTML5 audio 和 video 元素控件，创建基于浏览器的游戏（使用 HTML5 的 canvas 元素）。还可以利用一些强大的 HTML5 特性及相关技术编写强大的 Web 应用程序（这属于高级主题，不在本书的范围之内）。

如你所见，JavaScript 的功能非常强大，而它的使用也呈现了爆炸式的增长。jQuery（jquery.com）、MooTools（mootools.net）、YUI（yuilibrary.com）等 JavaScript 库确保了为页面添加简单交互和复杂行为的过程变得容易了许多，同时也对实现跨浏览器行为一致性提供了帮助。在这些库中，jQuery 是用得最多的一个，这主要是因为初学者很容易上手，同时它有很好的在线文档和大型社区支持。

浏览器厂商花费了大量的时间让它们的浏览器处理 JavaScript 的速度较几年前的版本有了显著的提升。JavaScript 也可以在平板电脑和现代移动浏览器中运行，不过出于性能方面的原因，可能需要为这些设备考虑在页面中加载的脚本的大小。

不过 JavaScript 本身是一个独立、庞大的主题，因此我们不会在本书讲解这门语言。我仍会解释如何将创建好的脚本插入到 HTML 文档中去，同时给出一些关于如何在插入脚本时尽量降低其对页面影响的建议，此外还会提供对事件处理程序的概览。

19.1 加载外部脚本

脚本主要有两种类型，一种是从外部文件（使用纯文本格式）加载的脚本，另一种是嵌入在页面中的脚本（这一种将在下一节讲解）。这同外部样式表和嵌入样式表的概念是类似的。

同为页面添加样式表一样，从外部文件加载脚本通常比在 HTML 中嵌入脚本要好一些（如图 19.1.1 所示）。这样做的好处也是类似的，即可以在需要某一脚本的每个页面加载同一个 JavaScript 文件。需要对脚本进行修改时，就可以仅编辑一个脚本，而不是在各个单独的 HTML 页面更新相似的脚本。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Loading an External Script</title>
<link rel="stylesheet" href="css/base.css" />
</head>
<body>
... All of your HTML content is here ...

<script src="behavior.js"></script>
</body>
</html>
```

图 19.1.1　script 元素的 src 属性引用脚本的 URL。在大多数情况下，最好在页面的最末尾加载脚本，即 </body> 结束标记的前面。也可以在页面的 head 元素中加载脚本，如图 19.1.2 所示，但这样做会影响页面显示的速率。更多信息参见“脚本和性能的最佳实践”

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Loading an External Script</title>
<!-- 在加载任何JS文件之前加载样式表 -->
<link rel="stylesheet" href="base.css" />
<script src="behavior.js"></script>
</head>
<body>
…… 所有HTML内容写在这里 ……
</body>
</html>
```

图 19.1.2　在这个例子中，脚本是在 head 中加载的。它位于 link 元素的后面，因此不会影响 CSS 文件先于脚本开始加载。关于为什么要尽可能地避免在 head 中加载脚本，参见“脚本和性能的最佳实践”

无论是加载外部脚本还是嵌入脚本，均使用 script（脚本）元素。

加载外部脚本的方法

输入 <script src="*script.js*"></script>，这里的 *script.js* 是外部脚本在服务器上的位置及文件名。应该尽可能地将脚本元素放在 </body> 结束标记之前（参见图 19.1.1），而不是放在文档的 head 元素里 (参见图 19.1.2)。

提 示　一个页面可以加载多个 JavaScript 文件，可以包含多个嵌入的脚本（参见图 19.2.1）。在默认情况下，浏览器会按照脚本在 HTML 中出现的顺序对它们进行加载（如果需要的话）和执行。关于为什么需要尽可能地避免加载多个脚本，参见“脚本和性能的最佳实践”。

提 示　不理解 JavaScript 的浏览器（数量很少）或用户禁用了 JavaScript 的浏览器会忽略 JavaScript 文件。因此要确保页面不依赖于 JavaScript 为用户提供对内容的可访问性和基本体验。

提 示　为了保持组织文件良好，通常将 JavaScript 文件放在一个子文件夹中（常用的名称包括 js、scripts 等）。src 属性需要反映这一点，就像指向某一资源的其他 URL 一样。例如，如果图 19.1.1 中的文件位于名为 assets/js/ 的文件夹内，则应输入 <script src="assets/js/behavior.js"></script>。（这只是一个示例，还有其他表示 URL 的方式，参见 1.7 节。）

提 示　图 19.1.1 和图 19.1.2 中的 JavaScript 文件名为 behavior.js，也可以指定其他有效的名称，只要以 .js 为扩展名。

提 示　从技术上说，为页面添加 JavaScript 还有第三种方式：内联脚本。**内联脚本**是直接在 HTML 中特定元素属性上指定的 JavaScript 片段。在这里提到这种方式是为了告诉你，应该避免使用它们，就像避免使用内联样式表一样。正如内联样式表将 HTML 和 CSS 混合在一起，内联脚本则将 HTML 和 JavaScript 纠缠在一起，这与将它们分开的最佳实践不符。

脚本和性能的最佳实践

关于脚本和页面性能最佳实践的详细讨论超出了本书的范围，不过这里仍会提到几个影响较大的要点。

最重要的一点，理解浏览器如何处理脚本是很有用的。随着页面的加载，在默认情况下，浏览器会按照脚本在 HTML 中出现的顺序依次对每个脚本进行下载（对于外部脚本）、解析和执行。在处理脚本的过程中，浏览器既不会下载该 script 元素后面出现的内容（哪怕是文本），也不会呈现这些内容。这称为**阻塞行为**（blocking behavior）。

这条规则对嵌入脚本和外部脚本都有效。可以想象，这会影响页面的呈现速率，影响的程度取决于脚本的大小和它执行的动作。

大多数浏览器都会这样做，因为 JavaScript 可能包含其他脚本所依赖的代码、立即生成内容的代码或对页面进行调整的代码。浏览器需要在结束呈现之前考虑所有这些问题。

那么如何避免这一点呢？消除 JavaScript 阻塞的最简单的方法就是将所有的 script 元素放置在 HTML 结束之前，即 </body> 结束标记的前面。随便花点儿时间浏览一下其他人的网站，不难发现脚本是在 head 元素中加载的。除了少数必须这样做的情况，通常认为这是一种过时的做法，应当尽可能地避免。（必须这样做的一种情况就是加载第 11 章所述的 HTML5 剃刀。）如果确实需要在 head 中加载脚本，也要将它们放置在所有加载 CSS 文件的 link 元素之后（这也是出于性能的考虑）。

另一种简单的加快脚本加载速率的方法就是将 JavaScript 放在同一个文件中（或尽可能少的一些文件中）并压缩代码。通常，压缩代码会去除换行、注释及额外的空格（以及其他同未压缩的代码存在差异的地方）。你可以想象一下只在一个很长的行内编写代码，而永远不敲回车键。

可以使用以下工具压缩脚本：

- ❑ Google Closure Compiler
  http://code.google.com/closure/compiler/（供下载的版本及文档）
  http://closure-compiler.appspot.com（在线版本）
- ❑ YUI Compressor
  http://developer.yahoo.com/yui/compressor/（供下载的版本及文档）
  http://refresh-sf.com/yui/（非官方的在线版本）

上述每个工具都能减小文件的大小，而压缩的结果则因脚本而异。通常，浏览器加载一个文件比加载两个文件（或更多的文件）要快一些，即便单个文件的大小比不同文件的总大小还要大一些（除非大得多）。

这是两种常用且强大的方法，但真正说起来，它们也只是一些皮毛的东西。关于脚本加载的方法及其优化，强烈推荐 Steve Souders 所著的 *Even Faster Web Sites*[①]（O'Reilly Media，2009）一书，以及他的网站 www.stevesouders.com。不过有言在先，Souders 的某些讨论相对技术一些。

19

① 中文版《高性能网站建设进阶指南》已由电子工业出版社出版。——译者注

19.2 添加嵌入脚本

嵌入脚本位于 HTML 文档之内，同嵌入样式表很相似。嵌入的脚本包含在 script 元素内，如图 19.2.1 所示。嵌入脚本并不是首选的方法（参见 19.1 节），但有时必须这样做。

```
<!DOCTYPE html>
<html lang="en">
<head>
     <meta charset="utf-8" />
     <title>Adding an Embedded Script</title>
<link rel="stylesheet" href="css/base.css" />
</head>
<body>
……所有HTML内容写在这里……

<script>
/*
JavaScript代码写在这里
*/
</script>
</body>
</html>
```

图 19.2.1 嵌入脚本没有 src 属性。相反，代码位于页面之内。如果要嵌入脚本，应该尽可能地在 </body> 结束标记之前这样做。虽然也可以在 head 中嵌入脚本（如图 19.2.2 所示），但通常出于性能原因而避免这样做

添加嵌入脚本的步骤

(1) 在 HTML 文档中，输入 <script>。

(2) 输入脚本的内容。

(3) 输入 </script>。

提示 每个脚本都按照它们出现在 HTML 中的顺序依次进行处理，无论是嵌入脚本还是外部脚本（参见 19.1 节）。

提示 尽管 script 元素必须有结束标记（</script>），但在有 src 属性的情况下，不能在该元素的开始标记和结束标记之间嵌入脚本（参见 19.1 节）。也就是说，<script src="behavior.js">Some function in here</script> 是无效的。对于任何 script 元素，要么仅通过 src 加载外部脚本，要么嵌入脚本并不含 src。

```
<!DOCTYPE html>
<html lang="en">
<head>
     <meta charset="utf-8" />
     <title>Loading an External Script</title>
<!-- 在加载任何JS文件之前加载样式表 -->
<link rel="stylesheet" href="base.css" />
<script>
/*
JavaScript代码写在这里
*/
</script>
</head>
<body>
……所有的HTML内容写在这里……
</body>
</html>
```

图 19.2.2 在这个例子中，嵌入脚本位于 head 中。它位于 link 元素的后面，从而让 CSS 文件更快地加载。关于为什么要尽可能地避免在 head 中嵌入脚本，参见“脚本和性能的最佳实践”

19.3 JavaScript 事件

在本章的引言中，我提到对 JavaScript 的深入讨论超出了本书的范围。不过，我仍然想让你大体了解一下 JavaScript 事件，从而对 JavaScript 能做什么有一些基本的认识。

可以编写 JavaScript 对特定的、预定义的事件（由访问者或浏览器触发）进行响应。下面的列表只是编写脚本时可用的事件处理程序的一些简单示例。HTML5 引入了大量其他的事件处理程序，其中很多都是与 audio 和 video 元素相关的。有的触屏设备也开始支持特殊的基于触摸操作的事件处理程序。

注意下面的列表中的“mouse”（鼠标）代表所有“指针设备”。例如，onmousedown

会在访问者使用电子笔、真正的鼠标或其他类似的设备时被触发。

- onblur：访问者离开先前获得焦点的元素（参见 onfocus）。
- onchange：访问者改变元素的值或内容。通常用于表单字段（关于表单，参见第 16 章）。
- onclick：访问者点击特定的区域或在元素（如链接）获得焦点时按下回车键。
- ondblclick：访问者双击特定的区域。
- onfocus：访问者选择、点击或用制表键将焦点移至特定的元素。
- onkeydown：在指定的元素上，访问者按下一个键。
- onkeypress：在指定的元素上，访问者按下并松开一个键。
- onkeyup：在指定的元素上，访问者在输入后松开一个键。
- onload：浏览器完成页面的加载，包括所有的外部文件（如图像、样式表、JavaScript 等）。
- onmousedown：在指定的元素上，访问者按下鼠标键。
- onmousemove：访问者移动鼠标指针。
- onmouseout：访问者在鼠标指针停留的特定元素上移开鼠标。
- onmouseover：访问者将鼠标指向元素。
- onmouseup：访问者在点击元素后松开鼠标键（与 onmousedown 相反）。
- onreset：访问者点击表单的重置按钮或在该按钮获得焦点时按下回车键。
- onselect：访问者选择元素中的一个或多个字符。
- onsubmit：访问者点击表单的提交按钮或在该按钮获得焦点时按下回车键。

HTML5 事件处理程序的完整列表参见 http://dev.w3.org/html5/spec-author-view/global-attributes.html。一些触屏设备（如智能手机、平板电脑）包含的与触摸相关的事件处理程序包括 touchstart、touchend、touchmove 等（https://dvcs.w3.org/hg/webevents/raw-file/tip/touchevents.html）。

第 20 章

测试和调试网页

本章内容

- ❑ 尝试一些调试技巧
- ❑ 检查常见错误：一般问题
- ❑ 检查常见错误：HTML
- ❑ 检查常见错误：CSS
- ❑ 验证代码
- ❑ 测试页面
- ❑ 当图像不显示时
- ❑ 仍然有错误

你编写好了全新的页面，却发现它们在浏览器中并没有像你预期的那样显示，或者完全不显示，或者在你的默认浏览器中显示得很好，但客户使用其他的浏览器打开时却发现页面显得有些奇怪。

在 HTML、CSS 和众多的浏览器（尤其是旧浏览器）和平台之间，很容易产生各种各样的问题。本章会提示一些常见的错误，并帮助你清除自己造成的错误。

这些调试技术中的一部分看起来相当基础，但网页的问题往往也非常基础。在查找大的问题之前，要确保没有任何小的问题。我将在第一节中介绍如何进行检查。

确认代码正确以后，应该在一个或多个平台的几种浏览器中对网站进行充分的测试，查看各个页面是不是按照预期的方式工作（参见 20.6 节及“应该在哪些浏览器中进行测试”）。

20.1 尝试一些调试技巧

下面是用于排查网页错误的一些真实技巧。

- ❑ 首先检查常见的错误。
- ❑ 细心观察，有条不紊。
- ❑ 逐步开展工作。做出小的改动，并在每次改动后进行测试。这样才能在问题出现之后定位到问题的来源。
- ❑ 调试时，从你能确保正确的地方开始。之后再逐一添加可能出问题的部分，每次添加完以后都在浏览器中进行测试，直到找到问题的来源为止。
- ❑ 用排除法寻找产生问题的代码片段。例如，可以注释掉一半代码，检查问题是否出于另一半代码，如图 20.1.1 所示。再在有问题的代码中注释掉更小的部分，直到找出问题。（参见 3.17 节和 7.2 节。）

```
...

.entry {
    border-right: 2px dashed #b74e07;
    margin: 0 .5em 2em 0;
}

.entry h2 {
    font-size: 1.25em;
    line-height: 1;
}

/*
.continued,
.entry .date {
    text-align: right;
}

.entry .date {
    line-height: 1;
    margin: 0 1em 0 0;
    padding: 0;
    position: relative;
    top: -1em;
}

.intro {
    margin: -5px 0 0 110px;
}
*/

.photo {
    float: left;
    height: 75px;
    width: 100px;
}

.photo a {
    margin: 0;
    padding: 0;
}

...
```

图 20.1.1 我注释掉了这段代码中间的一部分，以查看它是不是出错的原因。注意，很多 HTML 和 CSS 编辑器都包含代码高亮，即自动为元素、选择器等设置不同的颜色。这对调试是有帮助的。例如，如果将 CSS 属性名称打错了，编辑器就不会以预期的颜色显示它，表明它是无效的

- 留意输入错误。很多令人费解的错误是由简单的输入错误造成的。例如，在 HTML 中以某种方式拼写类名，却在 CSS 中使用了另一个名称。
- 在 CSS 中，如果不确定问题是出在属性上还是出在选择器上，可以试着在选择器上添加极其简单的声明，如 `color: red;`、`border: 1px solid red;` 等（如果 red 已经用于网页的设计，也可以选择一种不常见的颜色，如 pink）。如果元素变成红色，那么问题就出在属性上，否则问题就出在选择器上（假定不存在另一个特殊性更强的选择器，也不存在比当前选择器更靠后的选择器）。
- 休息一下。有时，在散步一小时之后工作 15 分钟比用这一小时苦坐在电脑前工作的收获还要大。有时，问题是在小憩时想到解决方案的。
- 使用一个或多个开发工具，直接在浏览器中对 HTML 或 CSS 进行修改测试。或者使用这些工具审查代码，试着对问题进行定位。（参见“浏览器开发工具”。）

浏览器开发工具

浏览器都包含调试工具，或者有作为插件的调试工具。很多工具的特性都是相似的。你会发现，一种反复用到的特性就是直接修改 CSS 或 HTML 并立即查看它们对页面的影响。这可以让你快速地对修改进行测试，满意之后再将它们包含到代码里去。

下面是各个浏览器中最常用的工具。

- Firefox：特别流行的插件 Firebug（http://getfirebug.com）。此外，Web Developer（http://chrispederick.com/work/web-developer/）是稍有区别的一种工具，但也非常好用；在同一个链接的页面上，也可以下载用于 Chrome 的版本。
- Internet Explorer：IE8+ 有内置的开发工具 Developer Tools（http://msdn.microsoft.com/en-us/ie/aa740478）对于 IE6 和 IE7，可以安装 Internet Explorer Developer Toolbar（www.microsoft.com/download/en/details.aspx?id=18359）。
- Opera：Dragonfly（www.opera.com/dragonfly/）。
- Safari：Web Inspector（http://developer.apple.com/technologies/safari/developer-tools.html）。

网上还有演示如何使用这些工具的文档和视频。

在 20.3 节和 20.4 节中分别有 Firebug 和 Web Inspector 的实例。

20.2　检查常见错误：一般问题

浏览器之间的差异可能是由一些不明显的浏览器漏洞造成的，也可能是由于使用新技术造成的，但更常见的则是出于一些简单的问题。每个人在从新手到专家的道路上难免会犯一些低级错误。例如，你有可能认为问题出在代码上，从而花了大量时间进行调试，最后才发现你修改的是一个文件，上传和查看的却是服务器上的另一个文件！

下列建议中的大多数适用于使用网站在服务器上的 URL 对网站进行测试的情形。

查检笼统的常见错误

- 根据 20.5 节的描述，对代码进行验证。这是一个很好的起点，因为这样就能排除代码语法相关的错误了。
- 确保已上传要测试的文件。
- 确保上传的文件的位置是正确的。
- 确保输入的 URL 与要测试的文件是对应的。如果要查看的页面是从另一个页面跳转过来的，确保链接中的 URL 与页面的文件名和位置是完全匹配的。
- 确保上传文件之前已经进行了保存（包括最新修改）。
- 确保上传了所有辅助文件（包括 CSS、图像、音乐、视频等）。
- 确保 URL 的大小写与文件名的大小写是完全匹配的。（这也是推荐全部使用小写字母的原因，这样做可以降低输入 URL 时产生错误的可能性——既针对网站开发人员，也针对访问者。）确保文件名中没有使用空格（应使用短横线代替空格分隔单词）。
- 如果在以前的测试中曾禁用某项浏览器功能（如 JavaScript 支持），确保重新启用这些功能。
- 确保问题不是出自浏览器本身。对于这一点，最简单的方法就是换个浏览

器对页面再测试一遍。

- 在接下来的两节里，我会介绍如何检查 HTML 和 CSS 中的常见错误。

20.3　检查常见错误：HTML

有时，问题出现在 HTML 中。

检查 HTML 常见错误的方法

- 很容易出现一两个输入错误，如图 20.3.1 所示。确保所有的拼写都是正确的，属性的值都是有效的，如图 20.3.2 所示。使用 HTML 验证器可以查出这类错误，从而能迅速地改正（参见 20.5 节）。

```
<img scr="woody.jpg" width="200px"
→height="150px" alt="Woody the cat" />
```

图 20.3.1　你能找出问题吗？我将 src 拼错了，还在 width 和 height 的值中使用了单位。如果你没有注意到这些错误，可以使用 HTML 验证器。它们可以标出这类输入错误，从而节省发现这些错误的时间

```
<img src="woody.jpg" width="200"
→height="150" alt="Woody the cat" />
```

图 20.3.2　在正确的版本中，src 的拼写是正确的，width 和 height 的值中也去掉了 px

- 留心元素的嵌套。例如，如果先开始 <p>，再使用 ，就要确保 结束标记位于最后的 </p> 之前。
- 如果重音字符或特殊符号没有正常显示，要确保文档 head 元素开始后有 <meta **charset="utf-8"** />语句（如果不使用 UTF-8，也可以使用其他恰当的字符编码），同时确保文本编辑器是使用与之相同的编码保存 HTML 文件的。如果还有问题，试着使用恰当的字符引用。
- 确保属性值是用直引号而不是曲引号包围的。如果属性值是用双引号包围的（这也是惯例），则属性值中可以有单引号，如图 20.3.3 所示。如果属性值本身包含双引号，则应使用字符引用，如图 20.3.4 所示。

```
<img src="jungle.jpg" width="325" height="275"
→alt="Llumi's jungle" />
```

图 20.3.3　如果属性值包含单引号，可以照常用双引号包围属性值

```
<img src="cookie-the-cat.jpg" width="250"
→height="200" alt="Cookie's saying,
→&quot;Enough!&quot;" />
```

图 20.3.4　如果属性值包含双引号，属性值里的双引用号就应使用字符引用

- 确保所有元素都有正确的开始标记和结束标记。同时，不要对空元素使用分开的开始标记和结束标记，如图 20.3.5 所示。（从技术上说，即便省略结束标记，或者对空元素添加结束标记，浏览器也能正确地显示，但最好还是严谨一些。）

```
<img src="jungle.jpg" width="325" height="275"
→alt="Llumi's jungle"></img>
```

图 20.3.5　不要对空元素（如 img）包含结束标记，HTML 验证器会将这个例子标记为错误

- 使用浏览器开发工具可以在浏览器解析文档结构之后对结构进行审查，如图 20.3.6 所示，将它与你预期的元素嵌套进行比较，如图 20.3.7 所示。这可以帮你快速定位标记格式不正确、元素未闭合或过早闭合等错误。

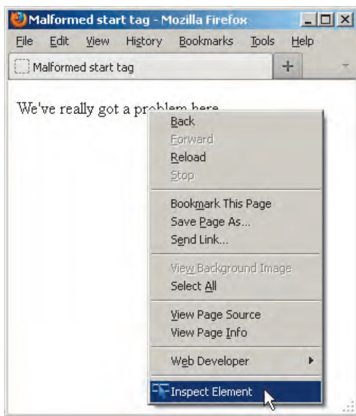


图 20.3.6　如果在 Firefox 中安装了 Firebug，可以右击内容（如果使用 Mac，则在按下 Control 键的同时单击），选择 Inspect Element。这时，就会在 Firebug 中显示内容的底层结构（参见图 20.3.7）

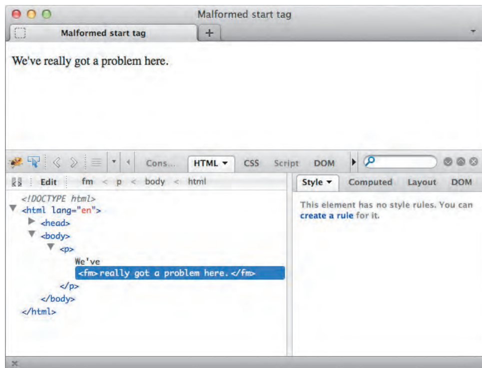


图 20.3.7　在 Firebug 中审查段落时，Firefox 在遇到 图 20.3.8 中有错的代码时会将 HTML 理解成 `<p>We've <fm>really got a problem here</fm>.</p>`

```
<p>We' ve <fm>really</em> got a problem here.
→ </p>
```

图 20.3.8　em 元素在开始标记中有一个输入错误。Firefox 等浏览器解析 HTML 时，会试着修复这个错误，并在显示页面时改变文档的底层结构，如图 20.3.7 所示。HTML 验证器也会将此标为错误

20.4　检查常见错误：CSS

尽管 CSS 的语法相当简单明了，但它也有一些常见的陷阱，特别是如果你习惯了编写 HTML 的话。CSS 验证器可以将如本节讨论的这类错误标示出来，因此，在查看 CSS 以寻找错误之前，要先对样式表进行验证。

检查 CSS 常见错误的方法

- 确保使用冒号（：）分隔属性和值，而不是像在 HTML 中那样使用等号，如图 20.4.1 和图 20.4.2 所示。

```
p {
    font-size=1.3em;
}
```

图 20.4.1　哎呀，改变用等号分隔属性和值的习惯很难

```
p {
    font-size: 1.3em;
}
```

图 20.4.2　好多了。属性和值之间应该使用冒号。在冒号之前或之后添加额外的空格不会产生影响，但通常在冒号之后加上一个空格

- 确保使用分号（；）结束每个属性 – 值对（即声明）。确保没有多余的分号，如图 20.4.3 和图 20.4.4 所示。
- 不要在数字和单位之间添加空格，如图 20.4.5 和图 20.4.6 所示。

```
p {
    font-size: 1.3em font-style: italic;;
    → font-weight: bold;
}
```

图 20.4.3　又错了。每个属性 – 值对之间必须有且只能有一个分号。这里，有一处少写了一个分号，另一处多写了一个分号

```
/* 还是有问题，但更容易发现问题了 */
p {
    font-size: 1.3em;;
    font-style: italic
    font-weight: bold;
}

/* 这是正确的版本 */
p {
    font-size: 1.3em;
    font-style: italic;
    font-weight: bold;
}
```

图 20.4.4 如果每个属性–值对都独占一行，错误就更容易发现了，因为这样分号就不会混在一大片属性、值和冒号之中了

```
p {
    font-size: .8275 em;
}
```

图 20.4.5 又错了。不要在数字和单位之间添加空格

```
p {
    font-size: .8275em;
}
```

图 20.4.6 这次对了。注意冒号和值之间的空格是可选的（但通常包含这个空格）

- ❑ 不要忘了后括号。
- ❑ 确保使用可接受的值。像 `font-style: none;` 这样的声明是无效的，因为该属性的空值为 `normal`。附录 B 为 CSS 属性和值的完整列表。
- ❑ 使用嵌入样式表时，不要忘了 `</style>` 结束标记。
- ❑ 确保 HTML 文档正确地指向 CSS 文件，且 URL 指向所需的文件。
- ❑ 留意选择器之间的空格和标点符号。
- ❑ 确保浏览器支持你编写的代码，尤其是 CSS3 代码，因为浏览器在 CSS3 成长的过程中也在不断演变。关于浏览器对特定属性和值的支持情况，参见附录 B 中提供的 URL。CSS 验证器无法判断浏览器是否支持某个 CSS 特性，但如果输入的选择器、属性或值是 CSS 中不存在的，验证器仍会给出提示。
- ❑ 使用浏览器开发工具审查浏览器所解析的样式规则以及当前计算出来的元素样式，从而快速地查出哪些代码没有按照预期进行解析，或者查看所应用的特殊性规则，如图 20.4.7 所示（参见“浏览器开发工具”）。

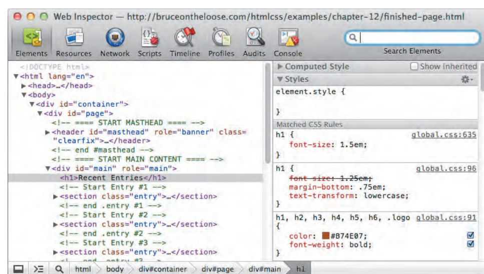


图 20.4.7 我使用 Safari 的 Web Inspector 对代码片段 `<h1>Recent Entries</h1>` 进行了审查。应用于该元素的 CSS 显示在右侧面板。`font-size` 设置上有一条删除线，表示这条声明被另一条规则（显示在上面）覆盖了。这样的结果是我在这个例子中想要的效果，但你可以使用这种技术跟踪样式没有按照预期进行应用的原因。还可以在右侧面板中对样式进行编辑，以测试不同的 CSS

20.5 验证代码

发现页面上的错误（如图 20.5.1 所示）的一种重要方法就是使用验证器，如图 20.5.2 所示。HTML 验证器可以对代码和语言规则进行比较，并将其发现的不一致的情况显示为错误或警告。它还可以提示语法错误，无效的元素、属性和值，以及错误的元素嵌套（如

图 20.5.3 所示）。它无法判断内容是不是由最能描述它的元素进行标记的，因此编写语义化的 HTML 还得靠你自己（参见 1.2 节）。CSS 验证器的工作原理也是类似的。

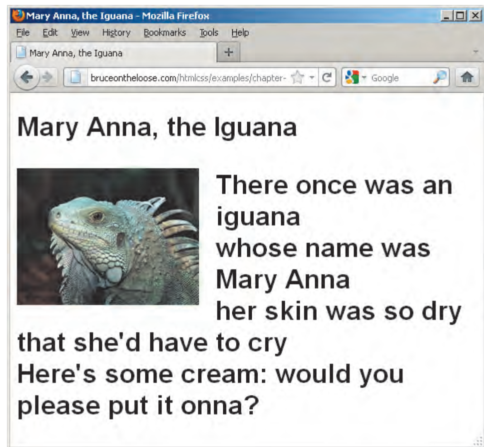


图 20.5.1　图像右边的文字本不该这么大。我已经检查了 CSS，问题不是出于设置了很大的 `font-size` 值。问题出在哪儿呢

将页面放到万维网上之前，不需要确保它们通过验证器的检查，完全没有错误。实际上，大多数网站都有一些错误。而且，W3C 的 CSS 验证器会将用于属性名称的厂商前缀标记为错误，但这并不意味着你应该将这些厂商前缀从样式表中移除（关于厂商前缀，参见第 14 章）。

浏览器可以处理很多类型的错误（同时忽略一些其他的错误），从而以它们能实现的最佳方式将页面呈现出来。因此，即便页面在验证时有错误，也可能看不出来。不过，有时错误会直接影响页面的显示（如图 20.5.1 所示）或行为。因此，应该使用验证器尽可能地排除代码中的错误。

验证器能找到的错误的示例参见 20.3 节和 20.4 节。

验证代码的步骤

(1) 首先使用 http://html5.validator.nu(参见图 20.5.2 和图 20.5.3) 或 W3C 的 http://validator.w3.org 对 HTML 进行检查。更多信息参见头两条提示。

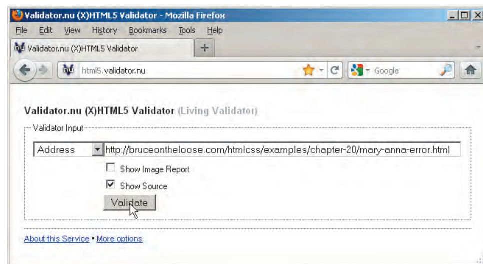


图 20.5.2　将需要检查的 URL 粘贴到 Address（地址）字段。选择 Show Source（显示源代码）选项（在默认情况下是未选中的），这样 HTML 源代码就会出现在验证器找到的错误的下方，有错误的 HTML 片段会突出显示

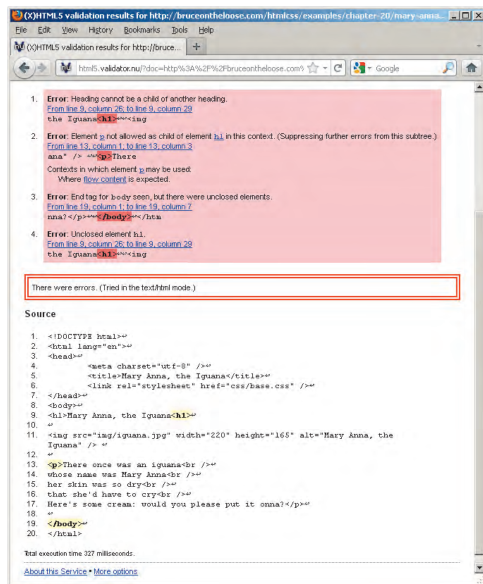


图 20.5.3　在第 9 行发现错误，原因是没有使用 `</h1>` 结束标记，错误地加上了另一个 `<h1>` 开始标记。其他的错误都是由第一个错误引起的，因此只要修复了这个错误，页面就没有任何错误了

(2) 修复标出来的 HTML 错误，保存修改。需要的话，将文件再次上传到服务器，再重复第 (1) 步。

(3) 可以使用 http://jigsaw.w3.org/css-validator/ 检查 CSS 错误。如果样式表包含 CSS3 特性，一定要在 Profile（配置）下拉菜单中选择“CSS level 3”（CSS 版本 3），如图 20.5.4 所示。否则，验证器会标示出比实际多的错误。

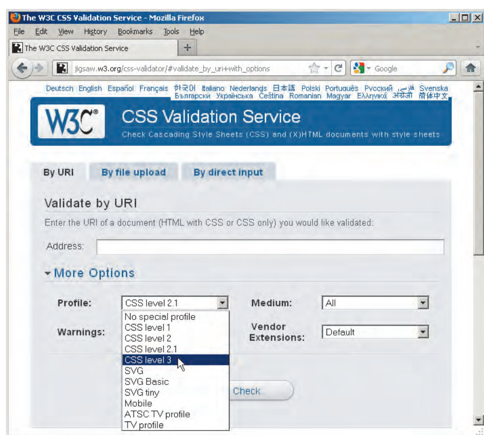


图 20.5.4　在默认情况下，验证器选择的配置为“CSS level 2.1”（CSS 版本 2.1）。如果样式表中包含 CSS3，则应选择“CSS level 3”。否则，验证器会显示更多的错误，因为 CSS 2.1 不包含 CSS3 的许多特性

提 示　W3C 的验证器（http://validator.w3.org/）使用的是 http://html5.validator.nu/ 提供的验证引擎，因此使用这两种都可以。W3C 的错误消息更易读，但不会对 HTML 源代码的错误部分突出显示。

提 示　可以通过输入 URL（参见图 20.5.1）、上传 HTML 文件以及将 HTML 粘贴到验证器三种方式对 HTML 进行验证。如果使用上传或复制粘贴的方式，不必将文件上传到服务器就可以进行检查了。

提 示　一个 HTML 错误可能导致多个验证器报错结果。例如，缺少一个结束标记会导致多条错误消息（参见图 20.5.3）。如果修复了这个结束标记的问题，所有这些后续错误就都不存在了。因此，应该按照从上往下的顺序，一次修复少量错误后就立刻再次验证，看看其他的问题是否也已解决。

提 示　HTML5 对代码的格式要求是很宽松的。例如，它并不关心像 img 这样的空元素的结束方式，因此 和 <img src="iguana.jpg" alt="Mary Anna, the Iguana"> 都是有效的。验证器无法判断代码在这些方面是不是一致的。如果你想确保代码的一致性，可以将 HTML 页面提交到 HTML Lint（http://lint.brihten.com/html/）。它可以检查空元素是否结束，开始标记和结束标记是否为小写字母，属性是否为小写字母，等等。

20.6　测试页面

即便代码通过了验证，页面可能仍然不像预期的那样工作，如图 20.6.1 所示，或者可能在一个浏览器中是正常工作的，在另一个浏览器中却有问题。在不同的浏览器和平台上对页面进行测试是很重要的（参见本节末尾的“应该在哪些浏览器中进行测试”）。

测试 HTML 页面

(1) 对 HTML 和 CSS 进行验证（参见 20.5 节），作出必要的修改。

(2) 打开浏览器，选择 File → Open File。找到要测试的网页，点击 Open。页面将出现在浏览器中。

(3) 检查整个页面，确保与你希望看到的

完全一样。例如：

- ❑ 格式与期望的是否一致
- ❑ 链接的 URL 是否指向了正确的页面或资源（可以通过激活链接并查看结果对 URL 进行测试）
- ❑ CSS 文件是否引用正确（参见图 20.6.1、图 20.6.2 和图 20.6.3）
- ❑ 所有的图像都出现了吗？它们的位置和对齐方式是对的吗？

(4) 在不关闭浏览器中页面的情况下，打开有关的 HTML 或 CSS 文档，作出必要的改动。

(5) 保存修改。

(6) 切换到浏览器，刷新页面并查看所作的改动。

(7) 重复第 (3) 步至第 (6) 步，直到你对网页满意为止。即便要测很多次，也不要感到沮丧。

再次对代码进行验证，确保没有引入任何其他的错误。

(8) 从第 (2) 步开始，在其他的浏览器中执行同样的测试流程，直到满意并认为页面做好了发布准备为止。

(9) 将文件上传至服务器。

(10) 回到浏览器，在地址栏中输入页面的 URL，按下回车键。页面将出现在浏览器中。

(11) 再次对页面进行检查，确保没有任何问题（现在，页面位于服务器上）。同时，如果访问者可能通过移动设备访问网站的话，别忘了在移动设备上对其进行检查。

提 示　推荐在将文件上传至服务器之前先对网站的本地版本进行完全测试。上传之后，针对服务器上的版本，再次对网站进行完全测试（不管在开发过程中对本地版本做过多少测试工作），因为这是访问者会看到的版本。

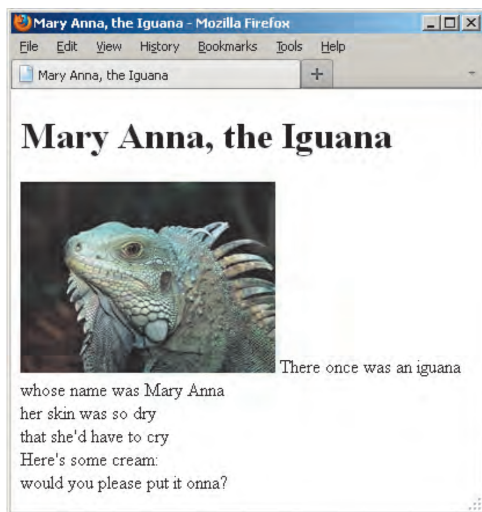


图 20.6.1　这个页面通过了验证，但它看上去并不像预期的那样。问题出在哪儿（参见图 20.6.2）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Mary Anna, the Iguana</title>
    <link rel="stylesheet" href=
    → "css/style.css" />
</head>
<body>

...

</body>
</html>
```

图 20.6.2　问题出在指向 CSS 文件的链接上——CSS 文件的名称为 Styles.css，而这里指向的却是 Styles.css。浏览器无法找到 CSS，因此对页面的显示是有问题的（参见图 20.6.1）。改正了代码中的文件名后，样式表就被加载进来了

提 示　再提醒一次，如果可能的话，要在多种平台上的多个浏览器中对 HTML 文档进行测试（参见“应该在哪些浏览器中进行测试”）。你无法预知访问者会使用哪种浏览器或计算机。

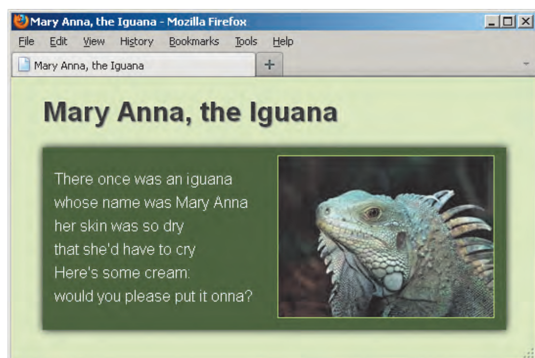


图 20.6.3 现在，指向 CSS 的链接改正了，页面显示正确

提 示 关于移动设备浏览器测试，参见12.3节。

提 示 如果浏览器中显示的是 HTML 代码而不是页面，要确保文件使用的是 .html 或 .htm 扩展名（而不是像 .txt 这样的扩展名）。

提 示 有时候，页面上的问题并不是你的错，尤其是样式上的问题。在认为问题出在代码上之前，一定要确保浏览器对遇到问题的特性是支持的。附录 A 和附录 B 中有关于 HTML 和 CSS 特性的浏览器支持情况的资源链接。

应该在哪些浏览器中进行测试

通常，大多数开发网站的人士会在以下浏览器中对网站进行验证：

- ❑ Chrome 的最新版本。Chrome 会在你的计算机上自动进行更新，大约每隔六周就会发布一个新的版本。Chrome 的下载地址为 www.google.com/chrome。
- ❑ Firefox 3.6+。Firefox 同 Chrome 一样，发行周期也很快，不过其更新并不是自动进行的。Firefox 在版本 3.6 之后已经有很多版本发布出来了，因此 3.6 不再是优先选择的版本了。Firefox 的下载地址为 www.firefox.com。
- ❑ Internet Explorer 7+。IE 的下载地址为 http://windows.microsoft.com/en-US/internet-explorer/downloads/ie。
- ❑ Safari 5+。在大多数情况下，对于该浏览器，只需要在其 Mac 版本中进行测试。尽管 Safari 也有 Windows 版本，但用户非常少，因此没有专门为此测试的必要。Safari 的下载地址为 www.apple.com/safari/。
- ❑ Opera 11+。Opera 在世界上很多地区都有少量的市场份额，不过它对 HTML5 的支持很好。Opera 的下载地址为 www.opera.com/。

由于 HTML5、CSS3、改进的 JavaScript 引擎以及其他技术的出现，浏览器的处理能力在近几年有飞速的增长。这里所列的大多数浏览器对 CSS 的呈现是相似的（大多数例外的情况都是关于 CSS3 的）。Internet Explorer 7 和 8 比其他所列的浏览器要旧得多，因此也更容易产生差异和错误。因此，如果你的网站在 IE7 和 IE8 中的样子同在现代浏览器中的样子相比有一点不同，也是可以接受的。

那么，如何对待 Internet Explorer 6 呢？由于 IE6 存在大量的怪异处理方式及错误，它已成为设计人员和开发人员多年以来的眼中盯。不过值得庆幸的是，它的市场份额已经大幅缩水了（参见 www.ie6countdown.com）。现在，网站拥有者为网站能在 IE6 中正常运转而付出额外精力的情况已经变少了，但是否这样做取决于网站的访问者。一些大的公司仍不愿完全

抛弃 IE6，但人们通常不再用它了，因此为 IE6 调整网站所花的大量精力不会带来太多收益。但是，了解你的访问者是必要的。在世界上某些地区（如亚洲的某些地方），还有大量的 IE6 用户。此外，一些大型组织也使用 IE6 作为其默认浏览器。

对所有浏览器和平台的用户都具有可访问性是很有挑战的。关于如何在各种浏览器（尤其是 Internet Explorer 的各个版本）上对页面进行测试的问题，参见 Addy Osmani 的文章（http://coding.smashingmagazine.com/2011/09/02/reliable-cross-browser-testing-part-1-internet-explorer/）。此外，如果你的朋友和家人使用你没有的浏览器，他们也可以帮到你。如果你的时间和资源有限，需要缩小测试的范围，也应该尽可能地在 Chrome 和 Firefox 的最新版本及 IE7+ 上进行测试。

浏览器市场的发展日新月异——你阅读到这一段话的时候，人们可能已经在试用这些浏览器更新的版本了。不过，只要遵循渐进增强的原则，你的网站就可以为旧的浏览器提供简单的体验，为现代浏览器提供增强的体验。

鉴于此，Yahoo! 推出了分级浏览器支持（Graded Browser Support）的概念（http://yuilibrary.com/yui/docs/tutorials/gbs/），并将这个概念应用于对其 JavaScript 和 CSS 框架 YUI（www.yuilibrary.com）的测试。该观点将浏览器分成不同的等级，不同的等级定义了对浏览器进行测试时的预期大小。你可以采取这种方法，在你的项目中对浏览器进行分级。

Google 对 Google Apps 采取的方法有所不同，该方法支持大多数浏览器的最新的两个版本（http://googleenterprise.blogspot.com/2011/06/our-plans-to-support-modernbrowsers.html）。同理，不同项目的需求可能不同。

20.7　当图像不出现时

小红叉、碎片图标、替代文本，或者什么都不显示——这都是图像未能正确加载的标志，如图 20.7.1 和图 20.7.2 所示。如果你希望显示的是一张大蜥蜴的照片，那么不管出现上述哪种情况都是令人恼火的。

修复缺失的图像

- 首先检查图像在服务器上的文件名是否与 img 元素中引用的名称严格匹配，包括大小写、扩展名（参见图 20.7.1）。
- 不要在文件名中包含空格。参见 1.6 节。
- 确保 img 元素的 src 属性中的图像 URL 是正确的。一种简单的测试方法就是将图像放到 HTML 页面所在的目录。这样就只需要 img 元素中的文件名和扩展名正确就可以了，无需引入路径信息。如果图像显示出来了，问题很可能就出在路径上。不过，将图像放在HTML文件所在的目录并不好，因为这样的话，网站很快就会变得无序。因此，在测试之后，要将图像从 HTML 页面目录移出，并修改指向图像的 src 路径。参见 1.7 节。
- 如果在你的计算机上查看页面时图像是显示的，而将页面上传至服务器之后，图像却没有显示出来，要确保已经将图像上传到了服务器。

- ❑ 把图像保存为 PNG、JPEG 或 GIF 格式了吗？如果是的话，所有的浏览器都能显示该图像；如果保存为 BMP 或 TIFF，则并非所有的浏览器都能显示图像。更多信息参见第 5 章。

```
...
<body>
<h1>Mary Anna, the Iguana</h1>

<p><img src="Iguana.jpg" width="220" height=
→"165" alt="Mary Anna, the Iguana" /> There
→once was an iguana ...</p>
</body>
</html>
```

图 20.7.1 图像的文件名为 iguana.jpg，但在 HTML 中，引用的却是 Iguana.jpg（以大写字母 I 开头）。因此，在服务器上检查页面时，图像不会显示（如图 20.7.2 所示）

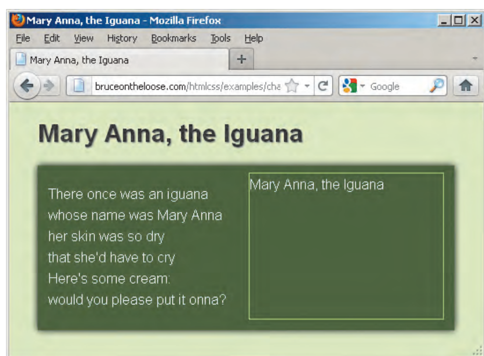


图 20.7.2 在你的计算机上，如果系统不区分文件名的大小写，页面看起来就没有问题。但将页面发布到区分大小写的服务器上以后，便无法找到图像，仅显示 alt 文本

20.8 仍然有错误

当我建议你休息一下的时候，不要认为我的建议是出于傲慢。有时候，你能做出的最佳选择就是暂时将问题放到一边。当你回头再进行处理时，答案可能就明显地摆在你面前。如果没有，不妨看看以下补充建议。

(1) 再次检查输入错误。再次对代码进行验证（参见 20.5 节）。

(2) 首先检查简单的错误。在查找问题的过程中，先检查你特别了解的地方，再研究不那么熟悉的地方。

(3) 简化问题。回到页面能正确工作的最近的版本。（为此，要在建立页面的过程中保存页面的副本，从而在必要的时候能回到历史版本。）再在每次一点一点地添加新元素的过程中对页面进行检查。

(4) 对于页面链接到的资源（如 CSS、图像、JavaScript 或媒体文件），直接在浏览器的地址栏中输入资源的 URL，确保该资源确实是存在的。

(5) 再次阅读本章——你可能在第一次漏掉了一些内容，或者某些内容可能启发你。

(6) 到网上搜索解决方案或指导意见。Stack Overflow（www.stackoverflow.com）和 SitePoint（www.sitepoint.com/forums/forumdisplay.php?40-design-your-site）便是两个例子。通过在网上搜索，还可以找到其他的资源。

第 21 章

发布网站

21

本章内容

- 获得域名
- 为网站寻找主机
- 将文件传送至服务器

当你完成你的杰作并准备好将它展现给公众时，必须将页面传送到 Web 主机服务器上，人们才能访问这些页面。

关于上传文件的最佳方式，你可以联系 Web 主机提供商或互联网服务提供商（Internet Service Provider，ISP）。通常，它们会提供一套指南，就如何连接到他们的服务器和哪里可以上传文件的问题提供说明。

在发布页面之前和之后，要确保对页面进行彻底的测试。更多细节信息参见第 20 章。

21.1　获得域名

在访问者能够看到你的网站之前，需要为网站关联一个域名，如图 21.1.1 所示。你可以注册自己的域名，并找一个 Web 主机存放网站，这样，任何人都可以在浏览器中访问这个域名（参见 21.2 节）。如果你决定换一个 Web 主机（或者主机提供商歇业了），可以让域名指向另一个 Web 主机的服务器，而所有的 URL 都保持不变。

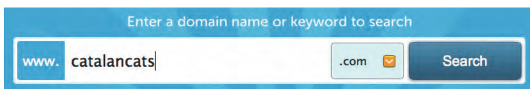


图 21.1.1　只有特定的公司才是可信的域名注册商（这个界面和下面的界面均来自 www.namecheap.com，选择此域名注册商没有代言之意）。你可以使用其中的一个查询你想使用的域名是否可用，也可以通过 Web 主机提供商的网站进行检查

获得域名的步骤

(1) 在浏览器中访问某个域名注册商（域名注册商的列表参见 www.internic.net/alpha.html），查看你想用的域名是否可用，如图 20.1.2 所示。（很多 Web 主机提供商也提供在其网站上查询域名是否可用的服务。）

(2) 一旦找到一个域名，便可以自行注册或通过你所使用的 Web 主机提供商代为注册（自行注册更为常见）。不同域名注册商的价格不一样，不过 .com 域名的价格通常大约为每年 10 美元（其他类型的域名的价格可能不一样）。有的 Web 主机提供商将域名注册作为主机服务费的部分折扣。

提示　要让网站在别人访问 URL 时显示出来，需要先进行一些重要的配置，这些内容参见下一节的“连接域名和 Web 主机”。

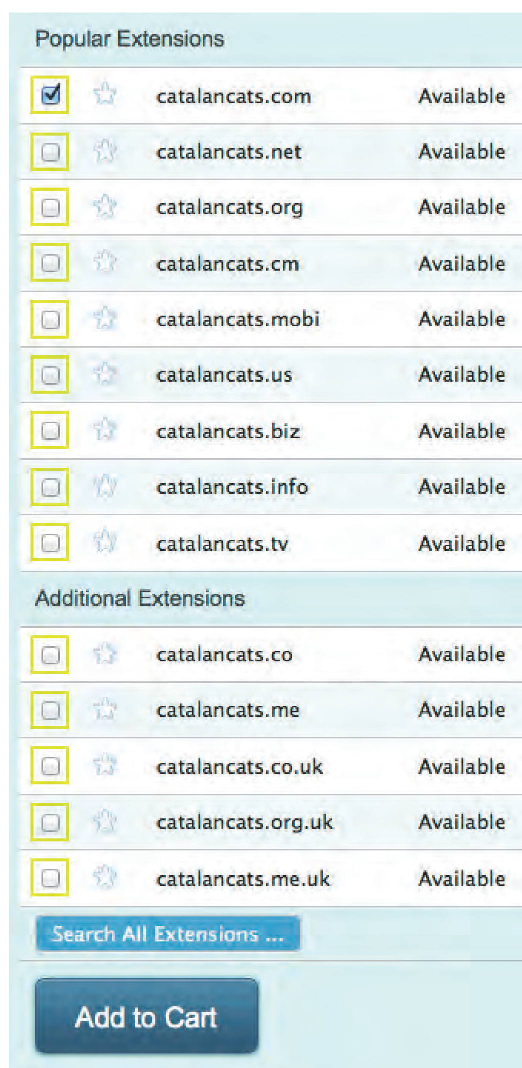


图 21.1.2 如果域名是可用的，就可以通过你查询域名的第三方注册商网站进行注册，或者通过 Web 主机提供商进行注册（从图中可以看到，www.catalancats.com 这个域名是可用的）

21.2 为网站寻找主机

除非你有自己的服务器，否则很有可能需要为你的网站寻找主机并支付一定的费用。Web 主机提供商可以提供其服务器上的一部分空间，存放你的网站文件，同时提供其他相关的服务，如创建同域名关联的电子邮件地址（如 *yourname@yourdomain*.com）。

有很多公司都提供网站托管服务。其中，大部分公司根据其服务的内容按月收取费用。有的公司提供免费的网站托管服务，但要求在你的网站上放置他们的广告。尽管你可以在互联网上搜索 Web 主机提供商，但还是推荐你向使用某种主机的朋友进行咨询——抑或某位值得信赖的博主提到的他所用的主机的提供商。

考虑主机提供商时，除了价格之外，还要考虑以下几件事件。

- 他们允许网站使用多大的磁盘空间？不要为超出需要的部分付费。话虽如此，通常连最基本的账户也拥有足够的空间。记住，HTML 文件占用的空间非常小，而图像、音频和视频文件则依次占用更大的空间。
- 他们允许每个月使用多大的数据传输量（带宽）。这个值代表的是发送给访问者的数据（包括 HTML、CSS、图像、媒体文件等）的总大小，而不是允许在服务器存储的空间的大小。因此，如果你预计访问者会从你的网站访问大量的大文件，就需要更大的月传输配额。
- 他们有应对大访问量、防止网站崩溃的措施吗？
- 可以使用域名创建多少邮箱（主机提供商通常提供很多个。）
- 账户可以为多个域名提供主机吗？是否需要为每个站点建立独立的账户？
- 他们提供哪种技术支持？是通过电话、邮件、还是在线交谈，他们响应请求需要多长时间，他们的网站上有

大量的支持信息吗？（在成为他们的客户之前，可以检查这些内容的数量。）

- 他们多久备份一次服务器上的数据（以防出现问题）？
- 可以使用哪些服务器端语言和软件包？他们使用 PHP、MySQL、WordPress 或其他高级特性吗？
- 他们提供 Web 分析报告吗？通过这样的报告，可以知道有多少人访问了你的网站及其他有用的数据。

将 ISP 作为 Web 主机提供商

如果你可以访问互联网，就可能已经通过你的 ISP 获得了少量的 Web 空间。它可能难以存放整个网站，但放几个页面倒是足够。可以向你的 ISP 询问详细信息。不过，要记住的是，这类托管空间通常不允许使用单独的域名访问网站，例如，可以使用 www.someisp.com/your-site/，但不能使用 www.yourdomain.com。因此，如果你希望建设的是专业的网站，就不要使用 ISP 提供的免费空间存放网站。

连接域名和 Web 主机

注册了域名，找到了 Web 主机之后，下面很重要的一步就是将二者结合到一起：你必须将你的域名指向你的 Web 主机，这样，访问者输入网站的 URL 时网站才能加载。

要实现这一步，需要对你的域名配置域名服务器。Web 主机提供商会提供用于这项配置的域名服务器信息。

根据注册域名的方式（参见 21.1 节）的不同，实际的配置是在以下两个地方中的一个进行的。如果域名是通过域名注册商注册的，可以登录到对应的账户，为域名设置域名服务器信息（域名注册商会提供操作说明）。如果域名是通过 Web 主机提供商注册的，可以登录到对应的账户，更新设置。

如果这些内容让你感到有些疑惑，也不必担心。Web 主机提供商和域名注册商会提供操作上的说明，通常，如果需要，他们还能提供手把手的帮助。

需要记住的是，当你修改域名服务器设置时，通常需要 24 至 48 小时（最多 72 小时）的时间，相应的更新才能传播到整个万维网。不过，这种改变不会在所有的地方同时发生。因此，如果你更新了域名服务器信息（并按照 21.3 节的说明上传了网站的文件），你的朋友可能已经可以在他所在的地方正常访问网站了，而你却无法立即看到网站（也可能是相反的情况）。你的网站应该在不久之后就能让所有人都看到。

21.3　将文件传送至服务器

为了让互联网上的人们看到你的页面，你需要将页面上传到 Web 主机服务器。一种简单的方法就是使用 FileZilla（http://filezilla-project.org）这样的 FTP 客户端。FileZilla 是一款自由软件，可以运行在 Windows、Mac OS X 和 Linux 上（关于其他的 FTP 客户端，参见提示）。很多网页编辑器也包含 FTP 功能，从而可以在编辑器中发布页面，而不必使用 FileZilla 这样的程序。

通常，你的 Web 主机提供商会在你注册托管账户之后通过电子邮件向你发送 FTP 连接信息。（如果你没有收到，可以联系他们。）一旦获得这些信息，就可以配置服务器连接，并将该配置保存在一个名称下面，如图 21.3.1、图 21.3.2 和图 21.3.3 所示，从而在以后想要发布文件（或从服务器上下载文件）的时候可以方便地访问服务器。

接下来，连接到服务器（如图 21.3.4 所示）和传输文件（如图 21.3.5 和图 21.3.6 所示）的操作相当简单。

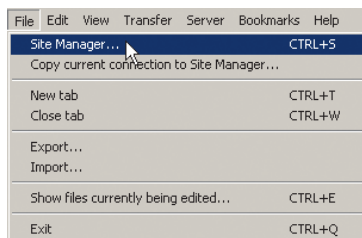


图 21.3.1　如果要在 FileZilla 中输入新的服务器信息，从主窗口中选择 File → Site Manager（文件→站点管理器）。Site Manager 是为每个站点配置 FTP 连接信息的地方

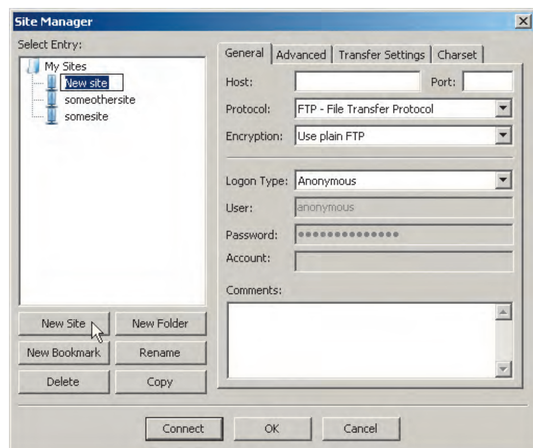


图 21.3.2　在 Site Manager 中点击 New Site（新站点）按钮，My Sites（我的站点）下面就会出现一个临时的名称

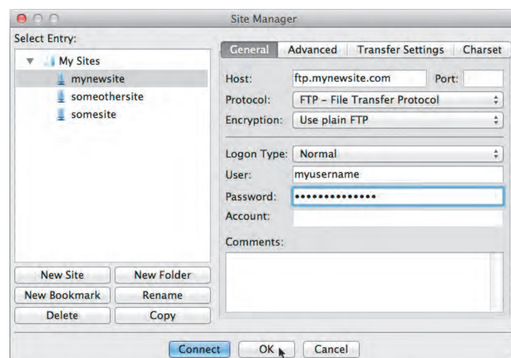


图 21.3.3　使用你选择的名称替换临时名称，再在 General（常规）标签页中配置连接信息。点击 Connect（连接）按钮，会保存这些信息并立即建立与服务器之间的连接。点击 OK（确定）按钮，则仅会保存信息

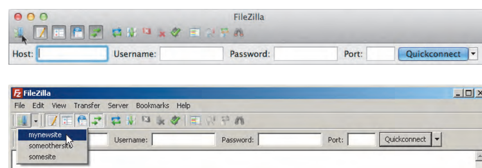


图 21.3.4　当站点的连接信息保存到 Site Manager 之后，就可以在不重复输入的情况下直接连接到 Web 主机的 FTP 服务器。在 Mac 或 Windows 上，回到 Site Manager（通过 A 中的服务器图标或菜单），从列表中选择你的站点，再点击 Connect（如图 21.3.3 所示）。在 Windows 上，还可以点击服务器图标（位于最左端）旁边的向下的箭头，再从弹出的下拉菜单中选择站点名称，如图所示。FiliZilla 会连接到服务器

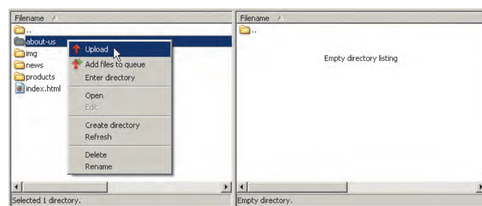


图 21.3.5　在窗口右侧，选择服务器的目标目录。在窗口左侧，定位到你的计算机上包含要上传的文件的目录。再在要上传至服务器的文件或文件夹上点击鼠标右键，选择 Upload（上传）

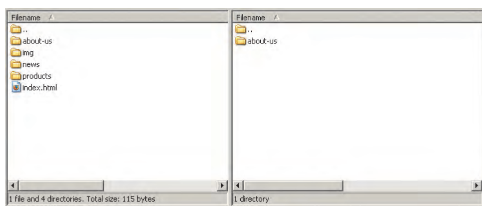


图 21.3.6　新上传的文件夹出现在窗口右侧的框内。对所有要传送至服务器的文件和文件夹执行相同的操作。如果要一次性上传多个文件和文件夹，可以先选中这些文件和文件夹，再点击鼠标右键并选择 Upload

注意，FileZilla 在 Mac OS 和 Windows 上的外观有些不同，但其界面是非常相似的（截图来自不同的操作系统）。除非特别指出，以下使用步骤对两个版本来说都是相同的。

1. 定义新的 FTP 站点的属性

(1) 在 FileZilla 的主菜单中选择 File → Site Manager（参见图 21.3.1）。

(2) 在 Site Manager 窗口中，点击 New Site（新站点）按钮（参见图 21.3.2）。My Sites 下面会出现一个临时名称。

(3) 输入站点名称（替换临时名称）。这个名称不必与域名相同，它只是一个标签。根据你的 Web 主机提供商的说明，填写 General 标签页下各字段的信息。通常，至少需要输入主机 URL，在 Logon Type（登录类型）中选择 Normal（常规），输入用户名和密码（通常在建立账号时创建），如图 21.3.3 所示。

(4) 输完连接信息之后，点击 Connect 按钮保存信息并立即连接到服务器，或者点击 OK 按钮保存信息供以后连接用，如图 21.3.3 所示。

2. 使用 FileZilla 传送文件至服务器

(1) 打开 FileZilla。

(2) 选择服务器图标（位于主菜单下方工具栏最左端）旁边的向下箭头，再在弹出的下拉菜单中选择你的站点名称，如图 21.3.4 所示。FileZilla 就会建立与服务器的连接。

(3) 在窗口的右侧，定位到要上传文件的服务器目录。

(4) 在窗口的左侧，定位到你的计算机上存放要上传的文件的目录。

(5) 在左侧框中要上传的文件或文件夹上点击鼠标右键，再在弹出的菜单中选择 Upload（参见图 21.3.5），文件就会进行传输，如图 21.3.6 所示（对于视频等大文件，这一过程所需要的时间会长一些）。还可以反过来传输文件（参见第一条提示）。

(6) 你对站点的更新现在已经能在线上看到了。通过 www.*yourdomain*.tld（这里的 *yourdomain*.tld 是你注册的域名；其中，.tld 是顶级域名，一般为 .com，但也可以注册使用其他扩展名的域名）访问你的站点，确保一切都是正常的。如果需要，编辑你的计算机上的文件，再按照第 (3) 步至第 (5) 步的说明将它们上传至服务器（如果过去了很长时间，则还需执行第 (2) 步）。重复这一步，直到网站完全符合你的预期。

(7) 完成文件传输之后，可以关闭 FileZilla 或在主菜单中选择 Server → Disconnect（服务器→断开连接），如图 21.3.7 所示。

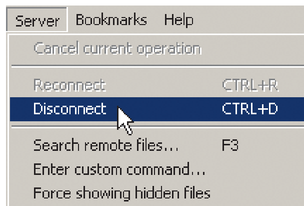


图 21.3.7　完成之后，选择 Server → Disconnect

提 示　还可以将文件从网站服务器传至你的计算机。要实现这一步，只需要右键单击右侧框中的文件或文件夹，再在弹出的菜单中选择 Download（下载）。

提示　FileZilla 只是众多 FTP 客户端中的一个。Mac OS X 上流行的其他 FTP 客户端还有 CyberDuck（自由软件，http://cyberduck.ch）、Transmit（www.panic.com/transmit）和 Fetch（http://fetchsoftworks.com）。Mac OS X 还有内置的 FTP 功能（参见 http://osxdaily.com/2011/02/07/ftp-from-mac/）。在网上搜索“FTP client”（FTP 客户端）可以找到更多的可以运行在 Windows 和 Mac 上的软件。它们的工作方式是相似的，只是有的软件功能多一些，有的少一些。

提示　传输文件和文件夹时，它们是被复制到目标文件夹的，原来的位置还保留它们的原始版本。

提示　如果要传输的文件或文件夹在目标位置已经存在，那么 FTP 程序可能会弹出对话框，确认是否要覆盖（FileZilla 就会这样做）。不过，每个 FTP 客户端都不相同，因此有的软件也可能不经确认而直接覆盖。可以试着传一个测试文件，了解你的 FTP 客户端是如何处理这类情形的。

提示　代码中的相对 URL 在文件夹传至服务器之后仍然有效。

提示　如果访问网站的 URL 时网站无法显示，可能有多种原因。首先，再次检查是否已将文件传至正确的目录。通常，页面应位于 *public_html*、*www* 或其他类似名称的目录下。你的 Web 主机提供商的操作说明中应该指明了正确的位置，如果不确定，可以向他们咨询。如果文件都放在正确的位置而网站依然无法显示，问题就可能出在域名服务器设置上（参见“连接域名和 Web 主机”）。

提示　如果将某文件的新版本上传至服务器之后却没有看到修改的效果，可以清除浏览器的缓存，并再次检查页面。如果不清楚如何清除缓存，可以搜索浏览器的帮助文档。

提示　对于大多数 FTP 客户端，可以通过改变窗口的大小一次性显示更多（或更少）的文件。要改变窗口大小，拖拽窗口的右下角即可。

附录 A

HTML 参考



本附录包含两个表格：表 A.1 和表 A.2。表 A.1 列出了全局属性，这类属性能够应用于大多数 HTML 元素；表 A.2 列出了 HTML 元素和其特殊属性（也就是非全局的）。这两个表中的大部分属性都在前文中讲过了。每个元素都有简单的描述，元素后面以缩进格式列出了与其相关联的属性及属性的注释文字。

HTML5 引入了大量新的元素和属性，还重新定义了一些 W3C 在早期的 (X)HTML 规范中不推荐使用的元素。对于来自 HTML5 的新的元素和属性，都在“版本”一栏中标上了“5”。对于在 HTML5 中有新的含义或额外含义的元素和属性，都标上了星号（*）。

要了解浏览器对 HTML5 特性的最新支持情况，请访问 http://caniuse.com/ 和 http://findmebyip.com/litmus。

A.1　HTML 全局属性

以下属性可应用于大多数 HTML 元素。

表 A.1　HTML 全局属性

全局属性	描　　述	版　本
accesskey	用于为元素添加键盘快捷键	
aria-*	用于关联由 WAI-ARIA 指定的可访问性属性值	5
class	用于标识一组元素，以便为它们应用样式	
contenteditable	用于让元素的内容变成可编辑的	5
contextmenu	用于标识元素的上下文菜单（其值必须与菜单元素的 id 属性值相同）	5
data-*	用于存储页面或应用特有的定制化数据	5
dir	用于指定元素的文字方向	
draggable	用于让元素变成可拖拽的	5
dropzone	用于将元素标记为可拖拽元素可以放下的区域	5
event	用于关联元素与脚本（event 表示某个实际的事件名称）	
hidden	用于指示元素还不是相关的或不再是相关的	5
id	用于标识特定的元素，以便为其添加链接，应用样式或使用 JavaScript 编写脚本	

（续）

全局属性	描　　述	版　本
lang	用于指定元素的书写语言	
role	用于为辅助设备提供关于元素角色的额外信息（由 WAI-ARIA 定义）	5
spellcheck	用于指示是否应该对元素的内容进行拼写和语法检查	5
style	用于添加本地样式表信息	
tabindex	用于定义访问者使用制表键时在元素之间移动的顺序	
title	用于为元素添加工具提示	

A.2 HTML 元素和属性

表 A.2 HTML 元素和属性

元素 / 属性	描　　述	版　本
a	用于创建链接和锚	
href	用指于定链接到的页面的 URL 或锚的名称	
hreflang	用于指定所链接的资源的语言	5
media	用于描述目标文档所定义的媒体类型	5
rel	用于标识链接的性质	
target	用于指定链接应打开的窗口或框架	*
type	用于指定资源的 MIME 类型	
abbr	用于解释缩写或首字母缩写的含义	*
address	用于为最近的 article 或 body 元素祖先标识联系人信息	
area	用于指定图像映射的坐标	
accesskey	用于为映射的特定区域添加键盘快捷键	
alt	用于给出关于区域的信息	
coords	用于给出图像映射中区域的坐标	
href	用于指定图像映射中区域链接的目标 URL	
hreflang	用于指定所链接的资源的语言	5
media	用于描述定义目标文档的媒体	5
rel	用于标识链接的种类	
shape	用于指定图像映射中区域的形状	
target	用于指定链接应打开的窗口或框架	*
article	用于标识页面中的独立成分，原则上是可独立分发或可再用的	5
aside	用于标识页面中的一个区域，其内容与周围的内容无关	5
audio	用于在页面中嵌入音频	5
autoplay	用于告诉浏览器在它能播放音频文件时立即开始播放	5
controls	用于告诉浏览器为音频元素提供控件	5
crossorigin	用于设置跨域请求凭证	5

（续）

元素 / 属性	描　述	版　本
loop	用于告诉音频文件在播放到末尾后不间断地继续从头播放	5
mediagroup	用于关联多个媒体文件	5
muted	用于控制音频输出的默认状态	5
preload	用于指定浏览器是否在访问者开始播放音频文件之前开始下载该文件	5
src	用于标识要播放的音频文件的 URL	5
b	用于标识出于实用的目的提醒读者注意的一块文本，不传达任何额外的重要性，也不表示其他的语态和语气	*
base	用于指定页面的基准 URL	
href	用于指定用于生成相对 URL 的 URL	
target	用于指定页面上链接的默认目标	*
bdi	用于标识独立于周围文本的用做双向文本格式化的一块文本	5
dir	用于指定文本方向	5
bdo	用于显式地格式化其内容的文本方向	
dir	用于指定文本方向	
blockquote	用于指定页面上的一段引述文本	
cite	用于给出来源的 URL	
body	用于包围页面的主要内容区域	
br	用于创建换行	
button	用于创建按钮	
autofocus	用于指定按钮在页面加载时立即获得焦点	5
disabled	用于指示元素在当前状态下是不可用的	
form	用于将元素同另外一个不包含该元素的表单关联起来	5
formaction	用于覆盖表单的 action 属性	5
formenctype	用于覆盖表单的 enctype 属性	5
formmethod	用于覆盖表单的 method 属性	5
formnovalidate	用于覆盖表单的 novalidate 属性	5
formtarget	用于覆盖表单的 target 属性	5
name	用于标识使用按钮发送的数据，或者用于标识按钮本身（或许是为了使用某项 JavaScript 功能）	
type	用于在表单元素中使用按钮	
value	用于指定在点击按钮时应该提交的数据	
canvas	提供用于生成依赖于分辨率的位图画布的脚本，以在线呈现图像	5
width, height	用于指定画布的尺寸	5
caption	用于创建表格的标题	
cite	用于将文本标记为引述	
code	用于将文本标记为计算机代码	
col	用于把表格中的列组合成非结构化组	

（续）

元素 / 属性	描　　述	版　本
span	用于指定列组中的列数	
colgroup	用于把表格中的列组合成结构化列组	
span	用于指定列组中的列数	
command	用于表示用户可以调用的命令，如定义键盘命令	5
checked	用于指示命令的选中状态（如果命令的类型为复选框或单选按钮的话）	5
disabled	用于指示命令在当前状态下是不可用的	5
icon	用于提供代表命令的图像	5
label	用于向用户显示命令的名称	5
radiogroup	用于标识当命令切换时它也跟着切换的单选按钮（如果命令的类型为单选按钮的话）	5
type	用于指定命令的类型	5
datalist	包含一组选项元素，这些元素是代表另一表单控件的一套预定义的选项	5
dd	用于标记列表中的定义	
details	用于创建公开的小部件，访问者可以通过它获取额外的信息或控制	5
open	用于指定元素在默认情况下是打开的还是关闭的	5
del	用于标记删除了的文本	
cite	用于引用对修订进行解释的 URL	
datetime	用于指定修订的时间和日期	
dfn	用于指定列表项目的定义实例	
title	用于提供术语的定义	
div	用于将页面切割为块级区域	
dl	用于创建定义列表	
dt	用于标记要在列表中定义的术语	
em	用于标记要强调的文本	*
embed	用于添加多媒体	*
src	用于指定多媒体文件的 URL	
type	用于标识多媒体文件的 MIME 类型	
width, height	用于指定嵌入的多媒体播放器的尺寸	
fieldset	用于将一套表单元素组合在一起	
disabled	用于将表单元素组内的所有表单控件设为不可用的	5
form	用于将元素同另外一个不包含该元素的表单关联起来	5
name	用于为表单元素组提供一个名称，供以后使用	5
figcaption	用于为其父元素 figure 的内容提供标题或说明文字	5
figure	用于识别在主文档流内被引用，但在不影响文档流的情况下可以移至他处的内容	5
footer	用于识别最近的祖先元素 body、section、article 或 aside 的页脚	5
form	用于指定表单，表单用于收集要提交的数据	

（续）

元素 / 属性	描　述	版　本
accept-charset	用于识别要在提交表单时使用的字符编码（默认为页面的字符集）	
action	用于给出处理表单数据的脚本的 URL	
autocomplete	当该属性设为 off 时，用于阻止浏览器提供或记住自动完成值（默认为 on，即在默认情况下允许自动完成）	5
enctype	用于确保文件以正确的编码格式发送至服务器	
method	用于指定数据应如何发送至服务器	
name	用于为表单提供名称，供以后使用	
novalidate	用于允许表单在不验证的情况下提交	5
target	用于识别表单提交的目标窗口或 iframe	*
h1，h2，h3，h4，h5，h6	用于创建标题	
head	用于创建 head 部分，该部分包含关于页面的信息，包括标题、制作者、关键词、样式表和脚本	
header	用于识别一组介绍性内容或导航帮助	5
hgroup	用于识别具有多个层级的区块标题	5
hr	用于标识段落级的主题变化	*
html	用于标识作为 HTML 文档的文本文档	
manifest	用于指定离线时使用的应用程序缓存清单	5
i	用于标记用另外一种语态和语气，或不同于常规方式陈述以表现不同特质的一块文字	*
iframe	用于创建嵌套的浏览上下文	*
name	用于指定作为目标的 iframe 的名称	
sandbox	用于出于安全目的，为 iframe 的内容指定额外的限制	5
seamless	用于指定 iframe 是否显示为包含它的页面的一部分	5
src	用于指定初始页面的 URL	
srcdoc	用于指定初始页面的 URL	5
width，height	用于指定 iframe 的尺寸	
img	用于在页面中插入图像	
alt	用于提供替代文本。当图像无法显示时，则显示替代文本；替代文本也是为使用辅助设备的用户提供的	
crossorigin	用于允许来自第三方站点（该站点允许跨域访问）的图像同 canvas 元素一起使用	
ismap	用于指示该元素可以提供对服务器端的图像映射（该元素必须是 a 元素的后代）的访问	
src	用于指定图像的 URL	
usemap	用于指定应同引用图像一起使用的客户端图像映射	
width，height	用于指定图像的尺寸，从而让页面的加载更快，或出于对图像进行缩放的目的	
input	用于创建表单元素	

（续）

元素 / 属性	描　述	版　本
accept	当输入框类型为 file 时，用于向浏览器告知需要接受的文件类型	
alt	当输入框类型为 image 时，用于提供替换文本	
autocomplete	当该属性设为 off 时，用于阻止浏览器提供或记住自动完成值（默认为 on，即在默认情况下允许自动完成）	5
autofocus	用于指定输入框在页面加载时立即获得焦点	5
checked	用于标记单选按钮或复选框在默认情况下被选中	
dirname	用于标识输入的文本的方向	5
disabled	用于指示输入框在当前状态下是不可用的	
form	用于将元素同另外一个不包含该元素的表单关联起来	5
formaction	用于覆盖表单的 action 属性	5
formenctype	用于覆盖表单的 enctype 属性	5
formmethod	用于覆盖表单的 method 属性	5
formnovalidate	用于覆盖表单的 novalidate 属性	5
formtarget	用于覆盖表单的 target 属性	5
list	用于将输入框与数据列表关联起来	5
max, min	用于指示输入框元素允许的值的范围	5
maxlength	用于指定可输入到输入框元素的字符的最大数量	
multiple	用于指定是否允许用户输入一个以上的值	5
name	用于标识元素收集的数据	
pattern	用于提供可对输入框元素的数据进行检查的正则表达式	5
placeholder	用于为数据输入提供提示	5
readonly	用于防止访问者修改特定的表单元素	
required	用于要求元素在提交表单时不能为空（当输入框类型为 hidden、image 或按钮类型时不可用）	5
size	用于指定文本框或密码框的长度	
src	用于指定活动图像的 URL	
step	用于控制允许输入的值的间隔大小和特殊性	5
type	用于指定表单元素的类型为文本框、密码框、单选按钮、复选框、隐藏字段、提交按钮、重置按钮、活动图像、日期 / 时间框、数字框或颜色框；用于从一系列值中进行选择；或用于输入电话号码、电子邮件地址或一组搜索词	*
value	用于指定表单元素的默认数据	
width, height	用于指定输入框的尺寸（仅在输入框类型为 image 时可以使用）	*
ins	用于标记对文档增加的内容	
cite	用于引用对修订进行解释的 URL	
datetime	用于指定修订的时间和日期	
kbd	用于标记用户输入	
keygen	用于生成公钥—私钥对	5

（续）

元素 / 属性	描　　述	版　本
autofocus	用于指定 keygen 元素在页面加载时立即获得焦点	5
challenge	用于生成与密钥对伴生的诘问	5
disabled	用于指示元素在当前状态下是不可用的	5
form	用于将元素同另外一个不包含该元素的表单关联起来	5
keytype	用于标识要生成的密钥对类型	5
name	用于标识收集的数据	5
label	用于为表单元素添加标签	
for	用于指定标签所属的表单元素	
form	用于将元素同另外一个不包含该元素的表单关联起来	5
legend	用于为表单元素组添加标签	
li	用于创建列表项目	
value	用于指定列表项目的初始值（当该元素为 ol 的子元素时）	*
link	用于指向外部样式表或其他外部资源	
href	用于指定资源的 URL	
hreflang	用于指定所链接的资源的语言	5
media	用于定义样式表的目标媒体类型和（或）媒体特性	
rel	用于标识链接种类	
sizes	用于标识引用图标的大小（仅在 rel 属性为 icon 时可以使用）	5
title	用于为替代样式表或其他资源添加标签	
type	用于指出资源的 MIME 类型（仅在链接类型不为 text/css 时需要使用）	
map	用于创建客户端图像映射	
name	用于对映射命名，从而使其可在以后被引用	
mark	出于引用的目的，对与另一个上下文相关的文本进行突出显示	5
math	用于在页面中嵌入 MathML	5
menu	用于包含命令列表	*
label	用于为菜单添加标签	5
type	用于标识所使用的菜单的种类：context、list（默认值）或 toolbar	5
meta	用于关联页面的各种元数据	
charset	用于标识页面本身的字符编码	5
content	用于添加关于页面本身的额外信息	
http-equiv	用于创建指向其他页面的自动跳转，设置默认脚本语言，声明字符编码	
name	用于标识关于页面的额外信息	
meter	用于表示在已知范围内的量度	5
high, low	用于指定量度为 high 或 low	5
max, min	用于标识允许指定的值的最大值和最小值	5
name	用于标识收集的数据	5

（续）

元素 / 属性	描　述	版　本
optimum	用于标识最优值	5
value	用于指定量表的当前值（必需的属性）	5
nav	用于标识页面的一块区域，该区域包含指向其他页面或页面内不同部分的链接	5
noscript	用于提供脚本的替代内容	
object	用于在网页中嵌入对象	
data	用于标识要嵌入的多媒体文件的来源	
form	用于将元素同另外一个不包含该元素的表单关联起来	5
name	用于标识对象（例如，对其编写脚本）	
type	用于指出对象的 MIME 类型	
typemustmatch	用于指示对象 data 属性中指定的资源的 MIME 类型必须与对象 type 属性中标识的 MIME 类型相同	
usemap	用于指示对象是否拥有相关联的图像映射	
width, height	用于指定对象框的尺寸	
ol	用于创建有序列表	
reversed	用于指定列表是否为反序（…，3，2，1）	5
start	用于指定第一个列表项目的初始值	*
type	用于指定每个列表项目开始的数字类型	*
optgroup	用于对 select 元素中的 option 元素进行分组，一组内的 option 元素位于同一个标签下	
disabled	用于指示元素在当前状态下是不可用的	
label	用于为选项组添加标签	
option	用于创建 select 或 datalist 元素中的单独的选项	
disabled	用于指示元素在当前状态下是不可用的	
label	用于指定选项如何出现在菜单中	
selected	用于标记空白表单中默认被选中的菜单选项	
value	用于指定菜单选项的初始值	
output	用于表示计算结果	5
for	用于创建计算结果与进入计算过程的值之间的显式关联	5
form	用于将元素同另外一个不包含该元素的表单关联起来	5
name	用于标识收集的数据	5
p	用于创建段落	
param	用于设置对象的属性	
name	用于标识属性的种类	
value	用于设置有名称的属性的值	
pre	用于表示一块预格式化文本	
progress	用于标识任务的完成进度	5

（续）

元素 / 属性	描　述	版　本
max	必须为大于 0 的有效浮点数（如果有的话）	5
value	必须为大于或等于 0 的有效浮点数（且小于或等于 max 属性的值，如果有该值的话）	5
q	用于引用来自另一来源的短文	
cite	用于给出引用源的 URL	
rp	用于在不支持旁注标记的浏览器中的旁注标记文本周围显示括号	5
rt	用于标记旁注标记文本	5
ruby	用于允许文本被旁注标记所标记	5
s	用于标识不再准确或不再相关的文本	*
samp	用于呈现某程序或计算系统的样本输出	
script	用于为页面添加“自动的”脚本	
async	用于影响脚本的加载和执行	5
charset	用于指定外部脚本所用的字符集	
defer	用于影响脚本的加载和执行	
src	用于引用外部脚本	
type	用于指定脚本所用的脚本语言（仅在脚本类型不为 text/javascript 时需要使用）	*
section	用于识别文档的区块	5
select	用于创建可以从一组选项中进行选择的表单控件	
autofocus	用于指定 select 元素在页面加载时立即获得焦点	5
disabled	用于指示元素在当前状态下是不可用的	
form	用于将元素同另外一个不包含该元素的表单关联起来	5
multiple	用于允许用户在菜单中选择一个以上的选择	
name	用于标识从菜单收集的数据	
required	用于标识用户必须选择一个选项才能提交表单（第一个 option 子元素必须为占位符或空值）	5
size	用于指定在初始状态下菜单中可见的项目数（同时用于将菜单显示为列表）	
small	用于呈现像条文细则等次要注释	*
source	用于在 audio 或 video 元素中标识替代媒体资源	5
crossorigin	用于设置跨域请求凭证	5
media	用于标识资源的目标媒体类型	5
src	用于标识要播放的音频或视频文件的 URL	5
type	用于指出资源的 MIME 类型	5
span	用于包围元素中无直接语义含义的内容	*
strong	用于标识元素内特别重要的内容	*
style	用于在页面中嵌入样式信息	

（续）

元素 / 属性	描　　述	版　本
media	用于标识样式表的用处	
scoped	用于仅对元素的父元素的后代应用样式	5
type	用于标识样式表的 MIME 类型（仅在样式类型不为 text/css 时需要使用）	*
sub	用于创建下标	
summary	用于标识 details 父元素内容的摘要、标题或说明文字	5
sup	用于创建上标	
svg	用于在页面中嵌入可缩放矢量图形	5
table	用于创建表格	
tbody	用于识别表格的主体部分；比之于头部（thead）和尾部（tfoot）	
td, th	分别用于在表格中创建普通单元格和标题单元格	
colspan	用于让单元格跨越多列	
rowspan	用于让单元格跨越多行	
scope	用于指定 th 应用于哪些行、列、行组或列组	
textarea	用于在表单中创建文本块输入区域	
autofocus	用于指定文本区域元素在页面加载时立即获得焦点	5
dirname	用于识别输入的文本的方向	5
disabled	用于指示元素在当前状态下是不可用的	
form	用于将元素同另外一个不包含该元素的表单关联起来	5
maxlength	用于指定可输入到 textarea 元素的字符的最大数量	
name	用于标识从文本块收集的数据	
placeholder	用于为数据输入提供提示	5
readonly	用于保护文本区域的内容	
required	用于要求元素在提交表单时不能为空	5
rows, cols	分别用于指定文本块的行数和列数	
wrap	用于指定在字段内容提交时使用软换行或硬换行	5
tfoot, thead	用于识别表格的尾部和头部	
time	用于指定日期和时间	5
datetime	用于为元素的文本所表达的时间或日期提供机器可读版本	5
pubdate	用于指定元素的祖先 article 或 body 元素的发布日期和时间	5
title	用于创建页面的标题（必须使用）	
tr	用于在表格中创建行	
track	用于为 audio 或 video 父元素指定外部计时文本轨道	5
default	用于指示默认轨道	5
kind	用于识别轨道为 subtitles（字幕）、captions（标题）、descriptions（描述）、chapters（篇章）或 metadata（元数据）	5
label	用于为轨道提供用户可读的名称	5

（续）

元素 / 属性	描　　述	版　本
src	用于标识轨道数据的 URL	5
srclang	用于标识轨道数据的语言	5
u	用于显示一段文本，作为虽然明确地呈现却不怎么准确的非文本注解	*
ul	用于创建无序列表	
var	用于标记作为变量的文本	
video	用于嵌入视频、电影和有说明文字的音频文件	5
autoplay	用于告诉浏览器在它能播放视频文件时立即开始播放	5
controls	用于告诉浏览器为视频元素提供控件	5
crossorigin	用于设置跨域请求凭证	5
loop	用于告诉视频文件在播放到末尾后不间断地继续从头播放	5
mediagroup	用于关联多个媒体文件	5
muted	用于控制音频输出的默认状态	5
poster	用于指定占位图片的 URL，该图片在媒体加载时或加载出现问题时显示	5
preload	用于指定浏览器是否在访问者开始播放媒体文件之前开始下载该文件	5
src	用于标识要播放的视频文件的 URL	5
width, height	用于指定视频的尺寸	5
wbr	用于识别在没有连字符的单词中可以在必要时进行换行的位置	5

附录 B CSS 引用

本附录包含以下 CSS 引用表格。

- 表 B.1　CSS 属性和值，可用做很多常见或实用的 CSS 属性及其默认值、允许值的快速参考。
- 表 B.2　CSS 选择器和结合符，对 CSS 选择器和结合符的引用，包括在 CSS3 中引入的那些。
- 表 B.3　CSS3 的颜色值，涵盖了在 CSS3 中引入的颜色值（HSL、HSLA 和 RGBA）。
- 表 B.4　CSS3 渐变，演示了在 CSS 中定义渐变的语法。
- 表 B.5　媒体查询，演示了媒体查询的语法。媒体查询用于对特定的媒体类型（如屏幕、打印）定位样式，以及对探测到的其他媒体特性（如设备尺寸、方向等）定位样式。
- 表 B.6　嵌入字体，演示了使用 `@font-face` 规则嵌入字体的语法。

注意：CSS3 规范还处于变化之中，浏览器也不断为规范中的各种模块更新额外的支持。浏览器厂商还引入了一些自己的新特性，或者实现了一些还未成为标准的 CSS3 特性。幸好，CSS 提供了一种安全的处理方式，即厂商前缀。厂商前缀允许不同的浏览器在不干扰其他浏览器的实现，且不覆盖未来标准版本的情况下，对属性进行实验性地实现（例如，使用 `-webkit-box-shadow` 而不是 `box-shadow`）。

最为常见的厂商前缀包括：

- `-moz-`（Firefox 及其他使用 Mozilla 渲染引擎的浏览器）
- `-webkit-`（Chrome、Safari 及其他使用 WebKit 渲染引擎的浏览器）
- `-o-`（Opera）
- `-ms-`（Internet Explorer 8 及以上版本）

关于哪些浏览器支持哪些 CSS3 属性和值的信息，可参考 www.quirksmode.org/css/contents.html、http://caniuse.com 及 http://findmebyip.com/litmus。此外，还可以使用像 Modernizr（www.modernizr.com）这样的 JavaScript 库测试浏览器对这些特性的支持情况。

B.1　CSS 属性和值

表 B.1　CSS 属性和值

属性 / 值	描述和注释
background 任何 background-attachment、background-color、background-image、background-repeat 和（或）background-position 值的组合，或 inherit	用于修改元素的背景颜色和背景图像 初始值取决于单独的属性，不继承的 background-position 可使用百分数 如果要显示多重背景，可使用逗号分隔组合背景值；如果要指定 background-color，应包含在最后一个背景中
background-attachment scroll、fixed 或 inherit	用于决定背景图像是否可以滚动，以及滚动的方式 初始值：scroll；不继承的 如果要显示多重背景，可以为每个背景应用不同的 background-attachment 值（用逗号分隔不同的值）
background-color 颜色值、transparent 或 inherit	用于设置元素的背景颜色 初始值：transparent；不继承的
background-image URL、CSS 渐变（参见表 B.4）、none 或 inherit	用于设置元素的背景图像 初始值：none；不继承的 如果要显示多重背景，可使用逗号分隔图像值
background-position 一个或两个百分数、长度（或一个百分数和一个长度），或 top、center、bottom 之一和（或）left、center、right 之一，或使用 inherit	用于设置指定的背景图像的物理位置 初始值：0% 0%；如果仅设置一个百分数，它会用于水平位置，而垂直位置的初始值则被设为 50%；如果仅使用一个关键字，另一个的初始值则为 center；应用于块级和替换元素；不继承的；百分数相对于盒本身的尺寸 如果要显示多重背景，可以为每个背景应用不同的 background-position 值（用逗号分隔不同的值）
background-repeat repeat、repeat-x、repeat-y、no-repeat 或 inherit 之一	用于确定背景图像是否重复及重复方式 初始值：repeat；不继承的 如果要显示多重背景图像，可以为每个背景图像应用不同的 background-repeat 值（用逗号分隔不同的值）
background-size 一个或两个百分数或长度，或 auto，或使用 cover 或 contain	用于指定背景图像的尺寸 初始值：auto；不继承的 如果要显示多重背景图像，可以为每个背景图像应用不同的 background-size 值（用逗号分隔不同的值）
border 任何 border-width、border-style 值和（或）颜色值的组合，或 inherit	用于定义元素四边边框的所有要素 初始值取决于单独的属性；不继承的
border-color 一至四个颜色值、transparent 或 inherit	用于指定元素的一个或多个边的边框的颜色 初始值：元素的 color 属性；不继承的
border-radius	用于为盒创建圆角 初始值：0；不继承的
border-top-right-radius border-bottom-right-radius border-bottom-left-radius border-top-left-radius	用于为盒的一个角设置 border-radius 值 初始值：0；不继承的 注：Firefox 的旧版本使用与此不同的语法创建单独的圆角：-moz-border-radius-topright、-moz-border-radius-bottomright、-moz-border-radius-bottomleft、-moz-border-radius-topleft

（续）

属性 / 值	描述和注释
border-spacing 一个或两个长度，或 inherit	用于指定表格边框之间的空隙大小 初始值：0；仅可应用于表格元素；继承的
border-style 一至四个以下的值：none、dotted、dashed、solid、double、groove、ridge、inset、outset、inherit	用于为元素的一个或多个边设置边框样式 初始值：none；不继承的
border-top、border-right、border-bottom、border-left 任何用于 border-width、border-style 的单个值和（或）颜色值的组合，或使用 inherit	用于为元素的一个边一次性定义全部三个边框属性 初始值取决于单独的值；不继承的
border-top-color、border-right-color、border-bottom-color、border-left-color 颜色值或 inherit	用于为元素的一个边定义边框颜色 初始值：color 属性的值；不继承的
border-top-style、border-right-style、border-bottom-style、border-left-style none、dotted、dashed、solid、double、groove、ridge、inset、outset 或 inherit	用于为元素的一个边定义边框样式 初始值：none；不继承的
border-top-width、border-right-width、border-bottom-width、border-left-width thin、medium、thick 或长度	用于为元素的一个边定义边框宽度 初始值：mdeium；不继承的
border-width 一至四个以下的值：thin、medium、thick 或长度	用于为元素的一个或四个边定义边框宽度 初始值：mdeium；不继承的
bottom 百分数、长度、auto 或 inherit	用于设置元素相对于其父元素底部边缘的位移大小 初始值：auto；不继承的；百分数相对于包含块的高度
box-shadow 可选的 inset，接着是二至四个长度值，接着是颜色值	用于为盒添加一个或多个阴影。长度值（依次）表示相对于盒右侧的位置（负数则表示相对于盒左侧的位置）、相对于盒底部的位置（负数则表示相对于盒顶部的位置）、模糊半径（不可为负数）和伸展距离（负数会让阴影收缩）。每个 box-shadow 值之间用逗号分隔 初始值：none；继承的
clear none、left、right、both 或 inherit	用于防止元素包围在浮动元素的一边或两边 初始值：none；只能应用于块级元素；不继承的
clip auto、rect 或 inherit	用于仅显示元素的一部分 初始值：auto；只能应用于绝对定位的元素
color 颜色值或 inherit	用于设置元素的文本颜色 初始值：父元素的颜色；有的颜色是由浏览器设置的；继承的
cursor auto、crosshair、default、pointer、progress、move、e-resize、ne-resize、nw-resize、n-resize、se-resize、sw-resize、s-resize、w-resize、text、wait、help、URL 或 inherit 之一	用于设置指针的形状 初始值：auto；继承的

（续）

属性 / 值	描述和注释
display inline、block、inline-block、list-item、run-in、compact、table、inline-table、table-row-group、table-header-group、table-footer-group、table-row、table-column-group、table-column、table-cell、table-caption、ruby、ruby-base、ruby-text、ruby-base-group、ruby-text-group、none、inherit 之一	用于确定元素如何显示，以及是否显示 初始值：通常为 inline 或 block；不继承的
float left、right、none、inherit 之一	用于确定元素向父元素的哪一边浮动 初始值：none；不可应用于定位过的元素[①]或生成的内容；不继承的
font 如果需要，任何 font-style、font-variant 和 font-weight 值的组合，接着是必需的 font-size、可选的 line-height 值和必需的 font-family，或使用 inherit	用于设置文本的字体系列、字体大小（这二者是必需的）及可选的字体样式、变体、粗细和行高 初始值取决于单独的属性；继承的；font-size 和 line-height 可使用百分数；font-size 和 font-family 是必需的，否则 font 属性是无效的
font-family 一个或多个由引号包着的字体名称，接着是可选的表示类属的字体名称，或使用 inherit	用于为文本选择字体系列 初始值：取决于浏览器；继承的
font-size 绝对大小、相对大小、长度、百分数或 inherit	用于设置文本的大小 初始值：medium；计算的值是继承的；百分数相对于父元素的字体大小
font-style normal、italic、oblique 或 inherit	用于将文本标记为斜体 初始值：normal；继承的
font-variant normal、small-caps 或 inherit	用于设置小型大写字母 初始值：normal；继承的
font-weight normal、bold、bolder、lighter、100、200、300、400、500、600、700、800、900 或 inherit	用于应用、移除、调整粗体格式 初始值：normal；数字值当做关键字而非整数进行处理（例如，不能使用 150）；继承的
height 长度、百分数、auto 或 inherit	用于设置元素的高度 初始值：auto；可应用于除了非替换行内元素、表格列和列组以外的任何元素；不继承的
left 长度、百分数、auto 或 inherit	用于设置元素相对于其父元素左侧边缘的位移大小 初始值：auto；只能用于定位过的元素；不继承的；百分数相对于包含块的宽度
letter-spacing normal、长度或 inherit	用于设置字母之间的间隙大小 初始值：normal；继承的

① 即设为绝对定位、相对定位或固定定位的元素。——译者注

（续）

属性 / 值	描述和注释
line-height normal、数字、长度、百分数或 inherit	用于设置文本行之间的距离 初始值：normal；继承的；百分数相对于元素自身的字体大小
list-style 任何 list-style-type、list-style-position 和（或）list-style-image 值的组合，或使用 inherit	用于设置列表的标识（常规的或定制的）及其位置 初始值取决于单独元素的初始值；只能应用于列表元素；继承的
list-style-image URL、none 或 inherit	用于为列表指定定制的标识 初始值：none；只能应用于列表元素；覆盖 list-style-type；继承的
list-style-position inside、outside 或 inherit	用于确定列表标识的位置 初始值：outside；只能应用于列表元素；继承的
list-style-type disc、circle、square、decimal、lower-roman、upper-roman、lower-alpha、upper-alpha、none 或 inherit	用于设置列表的标识 初始值：disc；只能应用于列表元素；如果 list-style-type 是有效的则不使用；继承的
margin 一至四个以下的值：长度、百分数、auto 或 inherit	用于设置元素与其父元素和（或）同胞元素之间在一个或多个边上的间隔大小 初始值取决于浏览器和 width 值；不继承的；百分数相对于包含块的宽度
margin-top、margin-right、margin-bottom、margin-left 长度、百分数、auto 或 inherit	用于设置元素与其父元素和（或）同胞元素之间在一个边上的间隔大小 初始值：0；不继承的；百分数相对于包含块的宽度；如果 width、margin-right 和 margin-left 之和大于父元素的包含块，则 margin-right 和 margin-left 的值会被覆盖
max-height、max-width 长度、百分数、none 或 inherit	分别用于设置元素的最大高度和（或）最大宽度 初始值：none；不能用于行内元素或表格元素；不继承的；百分数相对于包含块的高度 / 宽度
min-height、max-width 长度、百分数或 inherit	分别用于设置元素的最小高度和（或）最小宽度 初始值：none；不能用于行内元素或表格元素；不继承的；百分数相对于包含块的高度 / 宽度
opacity 0.0（表示完全透明）至 1.0（表示完全不透明）之间的任何小数	用于让元素半透明或不可见 初始值：1；不继承的
orphans 整数或 inherit	用于指定元素可以单独出现在页面底部的行数 初始值：2；只能用于块级元素；继承的；仅用于打印媒体
overflow visible、hidden、scroll、auto 或 inherit	用于确定当内容超出元素内容区域时额外的内容如何显示 初始值：visible；只能用于块级元素和替换元素；不继承的
padding 一至四个长度或百分数，或使用 inherit	用于指定元素内容区域和边框之间在一个或多个边上的距离 初始值取决于浏览器；不继承的；百分数相对于包含块的宽度
padding-top、padding-right、padding-bottom、padding-left 长度、百分数或 inherit	用于指定元素内容区域和边框之间在一个边上的距离 初始值：0；不继承的；百分数相对于包含块的宽度

（续）

属性 / 值	描述和注释
page-break-after、page-break-before always、avoid、auto、right、left 或 inherit	用于指定什么时候应出现分页，什么时候不应出现 初始值：auto；只能用于块级元素；不继承的；仅用于打印媒体
page-break-inside avoid、auto 或 inherit	阻止跨页的元素产生分页 初始值：auto；只能用于块级元素；继承的；仅用于打印媒体
position static、relative、absolute、fixed 或 inherit	用于确定元素如何相对于文档流进行定位 初始值：static；不继承的
right 长度、百分数、auto 或 inherit	用于设置元素相对于其父元素右侧边缘的位移大小 初始值：auto；只能用于定位过的元素；不继承的；百分数相对于包含块的宽度
table-layout fixed、auto 或 inherit	用于选择确定单元格宽度的算法 初始值：auto；不继承的
text-align left、right、center、justify、字符串或 inherit	用于指定文本对齐方式 初始值取决于浏览器和书写方向；只能应用于块级元素；继承的
text-decoration 任何 underline、overline、line-through 和 blink 的组合，或 none、inherit	用于修饰文本（大多数为线条） 初始值：none；不继承的
text-indent 长度、百分数或 inherit	用于设置段落第一行的缩进量 初始值：0；只能应用于块级元素；继承的；百分数相对于包含块的宽度
text-overflow clip、ellipsis 或 "string"	用于指定文本不可见时处理溢出的方式 初始值：clip
text-shadow 两个或四个长度值，接着是颜色值	用于为元素的文本添加一个或多个阴影。长度值（依次）表示相对于文本右侧的位置（负数则表示相对于文本左侧的位置）、相对于文本底部的位置（负数则表示相对于文本顶部的位置）、模糊半径（不可为负数）和伸展距离（负数会让阴影收缩）。每个 text-shadow 值之间用逗号分隔 初始值：none；继承的
text-transform capitalize、uppercase、lowercase、none 或 inherit	用于设置元素的文本的大小写 初始值：none；继承的
transform none 或一系列变形功能（matrix、translate、translateX、translateY、scale、scaleX、scaleY、rotate、skew、skewX、skewY）	用于对元素进行形状、大小或方向上的变形 初始值：none；不继承的；变形功能按照它们所列的顺序进行应用
transform-origin 一个或两个百分数或长度（或一个百分数和一个长度），或 top、center、bottom 之一和（或）left、center、right 之一	用于定义应用于元素的变形的起点 初始值：50% 50%；不继承的；只能应用于块级元素和行内元素；百分数相对于元素盒的大小

（续）

属性 / 值	描述和注释
transition 依次定义 transition-property、transition-duration、transition-timing-function 和 transition-delay 的简记法（用空格分隔）	用于为元素定义变形效果 初始值取决于单独的属性；可应用于所有的元素，包括 :before 和 :after 伪元素；值的顺序对此属性很重要
transition-property none、all 或用逗号分隔的一组 CSS 属性	用于识别在应用了变形的元素上定义的 CSS 属性 初始值：all；不继承的；可应用于所有的元素，包括 :before 和 :after 伪元素
transition-duration 以秒或毫秒为单位的时间值	用于确定完成变形所需的时间 初始值：0s（0 秒）；不继承的；可应用于所有的元素，包括 :before 和 :after 伪元素
transition-timing-function ease、linear、ease-in、ease-out、ease-in-out、cubic-bezier(number, number, number, number)	描述用于变形计算过程的中间值的使用方法 初始值：ease；可应用于所有的元素，包括 :before 和 :after 伪元素
transition-delay 以秒或毫秒为单位的时间值	用于定义变形开始的时间 初始值：0s（0 秒）；不继承的；可应用于所有的元素，包括 :before 和 :after 伪元素
top 长度、百分数、auto 或 inherit	用于设置元素相对于其父元素顶部边缘的位移大小 初始值：auto；只能用于定位过的元素；不继承的；百分数相对于包含块的高度
vertical-align baseline、sub、super、top、text-top、middle、bottom、text-bottom、百分数、长度或 inherit	用于指定元素在垂直方向上的对齐方式 初始值：baseline；不能应用于行内元素和表格单元格元素；不继承的；百分数相对于元素的 line-height 属性
visibility visible、hidden、collapse 或 inherit	用于在不将元素移出文档流的情况下让元素不可见 初始值：inherit，事实上是不继承的（仍存争议）
white-space normal、pre、nowrap、pre-wrap、pre-lined 或 inherit	用于指定如何处理空格 初始值：normal；只能用于块级元素；继承的
widows 整数或 inherit	用于指定元素可以单独出现在页面顶部的行数 初始值：2；只能用于块级元素；继承的；仅用于打印媒体
width 长度、百分数、auto 或 inherit	用于设置元素的宽度 初始值：auto；不能应用于行内元素、表格行或行组；不继承的；百分数相对于包含块的宽度
word-spacing normal、长度或 inherit	用于设置单词之间的距离 初始值：normal；继承的
z-index auto、整数或 inherit	用于设置元素相对于重叠元素的深度 初始值：auto；只能应用于定位了的元素；不继承的

B.2　CSS 选择器和结合符

表 B.2　CSS 选择器和结合符

模　　式	含　　义	CSS3	选择器类型
*	任何元素		通用选择器
E	类型为 E 的元素		类型选择器
E[foo]	带 "foo" 属性的 E 元素		属性选择器
E[foo="bar"]	"foo" 属性值恰为 "bar" 的 E 元素（引号是可选的）		属性选择器
E[foo ~ ="bar"]	"foo" 属性为一组空格分隔的值，且其中之一恰为 "bar" 的 E 元素（引号是可选的）		属性选择器
E[foo^="bar"]	"foo" 属性以 "bar" 开头的 E 元素（引号是可选的）	是	属性选择器
E[foo$="bar"]	"foo" 属性以 "bar" 结束的 E 元素（引号是可选的）	是	属性选择器
E[foo*="bar"]	"foo" 属性值在某处包含 "bar" 的 E 元素（引号是可选的）	是	属性选择器
E[foo|="en"]	"foo" 属性为一组连字符分隔的值且（从左边开始）以 "en" 开头的 E 元素（引号是可选的）		属性选择器
E:root	E 元素，文档根元素	是	结构伪类
E:nth-child(n)	E 元素，其父元素的第 n 个子元素	是	结构伪类
E:nth-last-child(n)	E 元素，其父元素的倒数第 n 个子元素	是	结构伪类
E:nth-of-type(n)	E 元素，该类型的第 n 个同胞元素	是	结构伪类
E:nth-last-of-type(n)	E 元素，该类型的倒数第 n 个同胞元素	是	结构伪类
E:first-child	E 元素，其父元素的第一个子元素		结构伪类
E:last-child	E 元素，其父元素的最后一个子元素	是	结构伪类
E:first-of-type	E 元素，该类型的第一个同胞元素	是	结构伪类
E:last-of-type	E 元素，该类型的最后一个同胞元素	是	结构伪类
E:only-child	E 元素，其父元素的唯一子元素	是	结构伪类
E:only-of-type	E 元素，该类型的唯一同胞元素	是	结构伪类
E:empty	没有子元素（含文本结点）的 E 元素	是	结构伪类
E:link, E:visited	作为目标尚未访问过（:link）或已经访问过（:visited）的超链接的 E 元素		链接伪类
E:focus, E:hover, E:active	处于特定用户操作下的 E 元素		用户操作伪类
E:target	作为引用 URI 目标的 E 元素	是	目标伪类
E:lang(fr)	语言为 "fr" 的 E 元素		:lang() 伪类

（续）

模　式	含　义	CSS3	选择器类型
E:enabled、E:disabled	状态为有效的或无效的用户界面 E 元素	是	UI 元素状态伪类
E:checked	选中了的用户界面 E 元素（如单选按钮或复选框）	是	UI 元素状态伪类
E::first-line	E 元素在格式上的第一行		::first-line 伪元素
E::first-letter	E 元素在格式上的第一个字母		::first-letter 伪元素
E::before	E 元素之前的生成内容		::before 伪元素
E::after	E 元素之后的生成内容		::after 伪元素
E.warning	类为 "warning" 的 E 元素		类选择器
E#myid	ID 等于 "myid" 的 E 元素		ID 选择器
E:not(s)	与简单选择器 s（例如 input:not(.warning)）不匹配的 E 元素	是	否定伪类
E F	作为 E 元素后代的 F 元素		后代结合符
E > F	作为 E 元素子元素的 F 元素		子元素结合符
E + F	紧接 E 元素后面的 F 元素		相邻同胞元素结合符
E ~ F	位于 E 元素后面的 F 元素	是	通用同胞元素结合符

表 B.2 是根据 www.w3.org/TR/css3-selectors/ 提供的 CSS3 选择器模型制订的，版权由万维网联盟（美国麻省理工学院、法国国家计算机科学与控制研究所、日本庆应义塾大学）所有。保留所有权利。

B.3　CSS3 颜色值

表 B.3　CSS3 颜色值

颜色值	描述和注释
rgb(red-value, green-value, blue-value)	RGB（红、绿、蓝）颜色模式 值可以是 0 到 255 之间的数字或百分数（不能是数字和百分数的组合） rgb(0, 0, 0) 和 rgb(0%, 0%, 0%) 为黑色 rgb(255, 255, 255) 和 rgb(100%, 100%, 100%) 为白色
rgba(red-value, green-value, blue-value, alpha)	RGB 颜色模式，加上 alpha 透明度 颜色值同 RGB 语法相同 第四个参数 alpha 是大于等于 0.0（完全透明）且小于等于 1.0（完全不透明）的小数
hsl(hue-value, saturation-value, lightness-value)	HSL（色相、饱和度、亮度）颜色模式 色相值用颜色环的角度（0 至 360 之间的数字）表示：0 和 360 为红色，120 为绿色，240 为蓝色，位于之间的其他值表示其他颜色 饱和度值用百分数表示：0% 为灰色，100% 为完全饱和的颜色 亮度值用百分数表示：0% 为黑色，100% 为白色，50% 为“正常”
hsla(hue-value, saturation-value, lightness-value, alpha)	HSL 颜色模式，加上 alpha 透明度 颜色值同 HSL 语法相同 第四个参数 alpha 是大于等于 0.0（完全透明）且小于等于 1.0（完全不透明）的小数

B.4　CSS3 渐变

CSS3 提供两种渐变样式—— linear-gradient（线性渐变）和 radial-gradient（径向渐变）。它们可以作为 background 和 background-image 属性的值。

表 B.4　CSS3 渐变

渐变样式 *	值	注　释
linear-gradient([origin,] color [stop], color [stop] [, color [stop]]*) 例如： linear-gradient(bottom left, #fff, #f00 30%, #000) 产生一个从盒的左下角向右上角过渡的渐变；从白色开始，在渐变路径 30% 处变为红色，最后变为黑色	origin 指定盒的角，可以是 top、left、bottom、right 和 center 关键字的组合或相对于盒尺寸的百分数值（从左上角开始） 第一个颜色值表示渐变开始的颜色值，最后一个颜色值是渐变结束的颜色值，可以在渐变中指定任意数量的颜色值 stop 指定颜色在渐变中的位置，可以是长度或相对于整个渐变长度的百分数	在默认情况下，线性渐变从盒的顶端中间部分开始 在默认情况下，浏览器会尝试在整个渐变中均匀地分布颜色 如果仅指定两种颜色，则默认的起始位置为 0% 和 100%
radial-gradient([origin,] [shape-or-size-or-both,] color [stop], color [stop] [, color [stop]]*) 例如： radial-gradient(30% 30%, circle closest-corner, #fff, #000) 产生一个从距盒的左上角 30% 远处开始，以圆形向周围辐射，直到到达最近的角为止的渐变；以白色开始（在圆圈的中心），以黑色结束（在外边缘）	origin 指定盒的角，可以是 top、left、bottom、right 和 center 关键字的组合或相对于盒尺寸的百分数值（从左上角开始） shape 在默认情况下为 circle 或 ellipse，这种形状会按照盒的大小进行填充（因此，当盒为矩形时，则使用椭圆形，当盒为方形时使用圆形） size 可以为以下关键字：closest-side、closest-corner、farthest-side、farthest-corner、contain、cover size 还可以使用长度值显式地设置径向渐变的尺寸（如果想分别设置水平长度和垂直长度，可以使用两个长度值） 第一个颜色值表示渐变开始的颜色值，可以在渐变中指定任意数量的颜色值 stop 指定颜色在渐变中的位置，可以是长度或相对于整个渐变长度的百分数	在默认情况下，径向渐变从盒的中心开始 在默认情况下，size 关键字会被设为 contain 在默认情况下，浏览器会尝试在整个渐变中均匀地分布颜色 如果仅指定两种颜色，则默认的起始位置为 0% 和 100%

* WebKit（Chrome 和 Safari 所用的渲染引擎）的早期版本使用的渐变语法与此并不相同。关于该语法更多信息参见 www.webkit.org/blog/1424/css3-gradients/。

B.5　媒体查询

表 B.5　媒体查询

特　　性	描述和注释
width、 min-width、 max-width 长度	输出设备的目标显示区域的宽度、最小宽度或最大宽度 应用：可视媒体和触觉媒体
height、 min-height、 max-height 长度	输出设备的目标显示区域的高度、最小高度或最大高度 应用：可视媒体和触觉媒体
device-width、 min-device-width、 max-device-width 长度	输出设备的呈现表面的宽度、最小宽度或最大宽度 应用：可视媒体和触觉媒体
device-height、 min-device-height、 max-device-height 长度	输出设备的呈现表面的高度、最小高度或最大高度 应用：可视媒体和触觉媒体
orientation portrait 或 landscape	当 height 特性值大于或等于 width 特性值时，方向为 portrait；否则为 landscape 应用：位图媒体
aspect-ratio、 min-aspect-ratio、 max-aspect-ratio 比例（如 4/3 或 16/9）	width 特性值与 height 特性值的比例、最小比例或最大比例 应用：位图媒体
device-aspect-ratio、 min-device-aspect-ratio、 max-device-aspect-ratio 比例（如 4/3 或 16/9）	device-width 特性值与 device-height 特性值的比例、最小比例或最大比例 应用：位图媒体
color、 min-color、 max-color 整数	输出设备每种颜色的位数、最小位数或最大位数；如果设备不是彩色的设备，则值为 0 应用：可视媒体
color-index、 min-color-index、 max-color-index 整数	输出设备颜色查询表中项目的数量、最小数量或最大数量；如果设备不使用颜色查询表，则值为 0 应用：可视媒体
monochrome、 min-monochrome、 max-monochrome 整数	在单色帧缓冲中每像素的位数、最小位数或最大位数；如果设备不是单色的设备，则值为 0 应用：可视媒体
resolution、 min-resolution、 max-resolution 分辨率（如 300dpi 或 118dpcm）	输出设备的分辨率、最小分辨率或最大分辨率（即像素密度）；resolution（不是 min-resolution 或 max-resolution）不会检测使用非方形像素的设备

（续）

特　　性	描述和注释
scan progressive 或 interlace	电视输出设备的扫描过程 应用：电视媒体
grid 0 或 1	设备是基于栅格的还是基于位图的；如果输出设备是基于栅格的（如 TTY 终端）则值为 1，否则值为 0；这种媒体查询也可以使用不带值的形式表示（例如 @media grid） 应用：可视媒体和触觉媒体

B.6　CSS 字体嵌入

表 B.6　CSS 字体嵌入（@font-face）

语　　法	描述和注释
@font-face {	开始样式规则
font-family: "name of font";	定义字体系列，可以用做 font-family 属性引用的值
src: url("path/to/font.ext") format("format-type");	识别字体的源文件；指定多个源文件时，使用逗号分隔各个值 对于 .eot 文件，格式应为 embedded-opentype；对于 .woff 文件，格式应为 woff；对于 .ttf 文件，格式应为 truetype；对于 .svg 文件，格式应为 svg
font-style: value;	指定嵌入的字体样式，使用与 font-style 属性相同的语法
font-weight: value;	指定嵌入的字体的粗细，使用与 font-weight 属性相同的语法
}	结束样式规则

延伸阅读

- JavaScript高级程序设计（第3版）
  978-7-115-27579-0，99.00
- HTML5程序设计（第2版）
  978-7-115-27871-5，59.00
- 响应式Web设计：HTML5和CSS3实战
  978-7-115-29922-2，49.00
- HTML5秘籍
  978-7-115-29018-2，79.00
- HTML5实战
  978-7-115-29752-5，59.00
- HTML5与CSS3设计模式
  978-7-115-29992-5，89.00
- HTML5和CSS3实例教程
  978-7-115-26724-5，39.00
- 深入HTML5应用开发
  978-7-115-27494-6，59.00
- HTML5 Canvas基础教程
  978-7-115-27101-3，49.00
- HTML5游戏开发
  978-7-115-26363-6，49.00
- 论道HTML5
  978-7-115-27870-8，69.00
- 精通CSS：高级Web标准解决方案（第2版）
  978-7-115-22673-0，49.00
- CSS3实用指南
  978-7-115-27378-9，49.00
- 精彩绝伦的CSS
  978-7-115-28479-2，49.00

“我们这些有过不少实际经验的设计师往往想当然地认为自己什么都知道，事实并非如此。本书告诉我们，我们知道的很多东西其实都是错的。所有Web设计师都需要看看这本书。”

——Web标准计划创始人Jeffrey Zeldman对本书上一版的评论

“Elizabeth Castro宝刀不老，而新作者Bruce Hyslop亦为本书增色不少。这本书既系统全面地讲解了基础知识，又能让读者快速吸纳HTML5和CSS3的绝大部分新特性，是一本值得拥有的好书！”

——Web Teacher评论

“快速入门和轻松写代码是这本书的最大特色，如果你不想浪费时间，而又想扎实地学习HTML5和CSS3，那就选这本书吧！”

——巴诺书店评论

“本书可谓字字珠玑，我入门时就是拿它当引路人的。”

——亚马逊评论

代表下一代网页编写技术的HTML5，为网页提供布局和格式的CSS3，这两者构成了Web开发的基石，也是Web程序员和设计师必须熟练掌握的最基本技能。

本书是风靡全球的HTML和CSS最佳入门教程的最新版，上一版单单英文版的销量就超过100万册，被翻译为十多种语言，并长期雄踞亚马逊书店计算机图书排行榜榜首。

最新的第7版秉承前一版直观、透彻、全面、循序渐进的讲授特色，仍然采用独特的双栏图文并排方式，手把手指导读者从零开始轻松入门。在内容上，第7版是一个重大的修订版本，全面反映了HTML5和CSS3的最新特色，让读者对网站设计领域振奋人心的进展感同身受。书中主要内容包括：如何创建HTML5页面，如何使用HTML5元素，如何用CSS3为网页添加样式，如何向页面添加JavaScript代码，如何测试做好的页面并将其上传到万维网。另外，书中还强调了渐进增强这种网站设计方法的重要性，并将其贯穿在全书的具体实践中。

作者专为本书设计了内容丰富的配套网站http://www.bruceontheloose.com/htmlcss/，提供书中全部示例的完整版本、勘误以及大量附加材料。


Peachpit Press



图灵社区：www.ituring.com.cn
新浪微博：@图灵教育　@图灵社区
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议　计算机/Web开发

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-30027-0
定价：59.00元